Probabilistic Methods for Coordination Problems

Habilitationsschrift

von

Christian Scheideler

Fachbereich Mathematik/Informatik und Heinz Nixdorf Institut

Universität Paderborn

Paderborn, März 2000

Reviewers:

- Prof. Dr. Friedhelm Meyer auf der Heide, Universität Paderborn, Germany
- Prof. Dr. Burkhard Monien, Universität Paderborn, Germany
- Prof. Dr. Hans Jürgen Prömel, Humboldt-Universität zu Berlin, Germany
- Prof. Dr. Bruce Maggs, Carnegie Mellon University, USA

Für Gabriele und
Klein-Christian

Preface

The problem of efficient coordination is at the heart of many problems in the area of computer science. In principle, coordination is necessary in any problem in which there are conflicting interests. We will concentrate on coordination problems such as

- assigning limited resources to requesting parties (e.g., routing problems),
- choosing orders in which to execute tasks (e.g., job shop scheduling problems),
- optimizing an objective under a collection of restrictions (e.g., combinatorial optimization problems), or
- fulfilling a collection of objectives (e.g., coloring or satisfiability problems).

In general, strategies for coordination problems may be classified in three different categories.

- *Global versus local*: Coordination may be performed globally, or locally in small (weakly dependent) parts of the problem.
- *Offline versus online*: In some cases, complete information about the coordination problem may be available. For others only a small part of the problem may be known beforehand.
- *Centralized versus distributed*: For some coordination problems it is possible (or even necessary) to develop efficient distributed algorithms for their solution. This may be of great importance if the coordination problem is huge. However, centralized algorithms have the advantage that they are often much simpler and therefore easier to implement.

In recent years it has turned out that probabilistic techniques are very powerful in solving coordination problems. There are many reasons for this. First of all, randomized algorithms are often much simpler and faster than their deterministic (or derandomized) counterparts. One of the reasons for this is that it is often much easier to avoid bad inputs by randomized strategies than by deterministic strategies. Furthermore, randomization often allows us to first solve a hard coordination problem in a relaxed form that allows us to find optimal solutions in an efficient way, and afterwards transform this solution into a (nearly optimal) solution for the original problem. (Consider, for instance, the widely used randomized rounding technique pioneered by Raghavan and Thompson [RT87].) Moreover, as we will show, randomization can be efficiently used to break a global coordination problem into many small coordination problems that can be treated (almost) independently. This does not only help to speed up the sequential time necessary to find a good solution, but it also enables us to construct efficient distributed algorithms.

Apart from the algorithmic aspect, it has turned out that probabilistic methods can also be used to prove powerful results about the existence of solutions to certain coordination problems.

This was pioneered by Erdős [Erd47] already back in 1947. Today, probabilistic proof methods are widely recognized as one of the major techniques in the field of combinatorics.

Probabilistic methods for both algorithms and proofs often require the use of tail estimates for sums of random variables. In an influential paper, Hoeffding [Hoe63] presented several bounds that enabled the analysis of many randomized algorithms. Since then, many authors have provided bounds not only for sums of random variables, but also for martingales, Markov chains, differential equations, and so on, that turned out to be very useful for many areas in computer science.

The purpose of this thesis is to give examples of how to apply probabilistic methods to coordination problems, concentrating on problems in the area of routing, job shop scheduling, hypergraph coloring, and certain combinatorial optimization problems. The text is self-contained with respect to the historical background and notations used. However, it is recommended that the reader has some experience with using probabilistic arguments in proofs. In the following, we give a short outline of the thesis.

Chapter 1 gives examples of coordination problems that will be important for this thesis and contains a brief introduction to the area of probabilistic techniques for coordination problems. Chapter 2 introduces the basic notation and summarizes the main results of this thesis.

In Chapter 3, we review the basic concepts in the area of probability theory and probabilistic techniques. Some of these results are based on my own research (such as parts of the section about witness structures and the section about the Lovász Local Lemma). Even if the reader already has some experience with using probabilistic arguments, it might be worth having a look at this chapter, since several of its proof techniques and results will be used later.

The following chapters give several examples of how to solve coordination problems both algorithmically and in an existential way. Concerning the design of algorithms, the first three chapters mostly concentrate on providing distributed online algorithms, whereas the last two chapters mostly concentrate on providing centralized offline algorithms.

Chapter 4 investigates how static store-and-forward routing algorithms can be transformed into efficient dynamic algorithms, that is, how algorithms that have been designed for the case that all packets are injected at the same time can be adapted to more realistic scenarios in which packets are continuously injected into the network. We will present a transformation scheme applicable to a large class of routing algorithms.

In Chapter 5, we study the problem of developing static and dynamic routing strategies that ensure that the runtime of a packet only depends on local parameters, such as the length of its path and the maximum number of other packets using an edge of its path, even if these parameters are *not* known in advance.

Chapter 6 studies the problem of routing in ad hoc networks. An ad hoc wireless network is a collection of wireless mobile hosts forming a temporary network without the aid of any established infrastructure or centralized administration. We concentrate on finding schemes for routing arbitrary permutations in these networks. In general, it is NP-hard even to find a $n^{1-\epsilon}$-approximation for any constant ϵ to the fastest possible strategy for routing a given permutation problem on n mobile hosts. However, we demonstrate that if we allow ourselves to consider slightly less general problems, efficient solutions can be found.

Chapter 7 considers coordination problems in the area of job shop scheduling. We prove nearly matching upper and lower bounds for the acyclic job shop scheduling problem. The results are of purely existential nature, that is, no coordination strategies are provided (that run in polynomial time).

Chapters 8 and 9 consider the problem of how to transform existential results based on probabilistic arguments into randomized polynomial time algorithms for constructing the respective results. These algorithms have in common that they aim to split a coordination problem into many small subproblems that can be solved independently.

In Chapter 8, we construct and analyze polynomial time algorithms for the problem of 2-coloring non-uniform hypergraphs. A hypergraph is called 2-colorable if its nodes can be colored by two colors in such a way that every hyperedge has a node of each color.

Chapter 9 presents a general method of turning existential results (that are based on the so-called Lovász Local Lemma) into polynomial time algorithms. The main emphasis will be on constructing polynomial time algorithms for certain combinatorial optimization problems called *minimax* integer programs (or MIPs). We present two applications of our method: first, we show how to transform the existential job shop scheduling results in Chapter 7 into polynomial time algorithms, and then we show how to construct good approximation algorithms for MAX SAT that are able to exploit non-uniform dependencies among the clauses of a given boolean formula.

A large part of this book has been previously published as joint work. The results in Chapter 4 are taken from [SV99], Chapter 5 is based on [BS99], and Chapter 6 comprises the results presented in [AS98]. Chapter 7 is based on [FS98], Chapter 8 contains the results in [CS00a], and Chapter 9 presents the results of [CS00b].

Acknowledgements

First of all, I would like to thank Prof. Dr. Friedhelm Meyer auf der Heide for his great support. Furthermore, I would like to express my thanks to those people with whom I had a great collaboration. To name them in alphabetic order, these are: Micah Adler, Petra Berenbrink, Artur Czumaj, Uriel Feige, and Berthold Vöcking. I would also like to thank all members of the research group Meyer auf der Heide for the good cooperation and the nice atmosphere.

Besides my colleagues, I am very grateful to my wife Gabriele whose continuous support greatly helped me to finish this thesis.

Paderborn, March 2000 *Christian Scheideler*

This work has been partly supported by

- DFG-Sonderforschungsbereich 376 "Massive Parallelität: Algorithmen, Entwurfsmethoden, Anwendungen",
- a fellowship from the MINERVA Foundation, and by
- EU ESPRIT Long Term Research Project 20244 (ALCOM-IT).

Contents

Chapter 1

Introduction

The problem of efficient coordination is at the heart of many problems in the area of computer science. In this thesis, we will concentrate on problems that require coordination due to conflicting interests. We start with some examples in this area that will be of importance in this thesis.

Example 1: Routing in Networks

Suppose that we are given a network and a set of source-destination pairs. For each of these source-destination pairs, we want to send a packet along the edges of the network from the source to the destination. See Figure 1.1 for a routing problem.

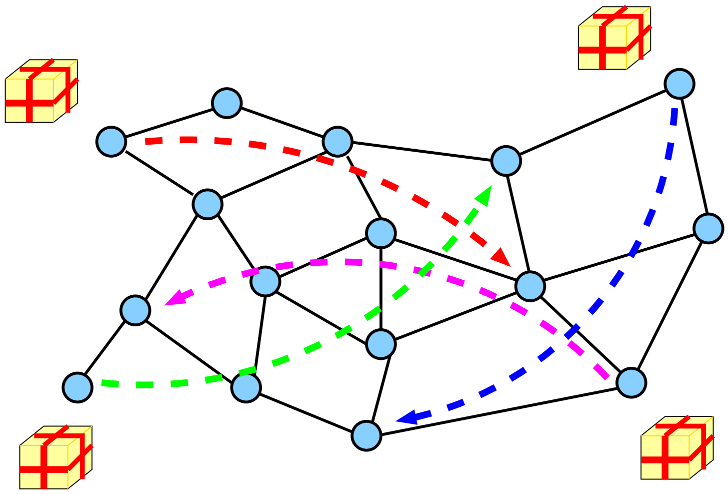

Figure 1.1: An example of a routing problem

Since the edges have a limited bandwidth (we usually assume that an edge can transport at most one packet per time step), two coordination problems have to be solved to ensure that all packets quickly reach their destinations:

- How to select paths for the packets?

- How to send the packets along their selected paths?

Problems of the first kind are called path selection problems, and problems of the second kind are called packet switching problems. Obviously, the path selection requires some coordinated strategy of the nodes, since otherwise all paths may end up using the same edge. In the case of packet switching, the problem has to be solved which packet to prefer if several packets contend to use the same edge at the same time. As we will see, it is often possible to use strategies for this problem that do not require any coordination among the nodes but nevertheless get quite close to the best possible performance of globally coordinated strategies.

Example 2: Job Shop Scheduling

A job shop scheduling problem is specified by a set of jobs and a set of machines. Each job is composed of a sequence of operations to be performed on different machines. Since it is usually assumed that a machine can be busy with at most one operation at a time step, the problem has to be solved which operation to prefer if several operations contend to be processed on the same machine at the same time step. Usually, the goal is to find an order in which to process the jobs by the machines, also called a *schedule*, such that the time necessary to complete all jobs is minimal. To give an example of a job shop scheduling problem, see Figure 1.2.

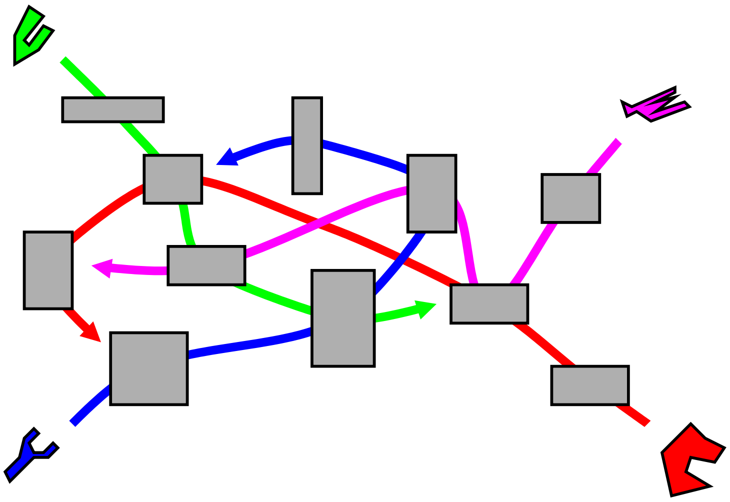

Figure 1.2: An example of a job shop problem.

Job shop scheduling problems can be seen as generalizations of switching problems. However, in contrast to the switching area, in the job shop scheduling area mostly strategies have been presented so far that require a global coordination among the machines.

Example 3: Hypergraph Coloring

Assume that we are given a hypergraph with hyperedges of size at least 2. The problem is to assign one out of two possible colors to each node of the hypergraph such that every hyperedge has at least one node of each color (see Figure 1.3 for a coloring that fulfills this property). Such a coloring is called a 2-coloring.

It is not difficult to see that there are hypergraphs that are not 2-colorable. Furthermore, it is NP-complete to decide whether a hypergraph has a 2-coloring or not. This indicates that it is by far not trivial to find a 2-coloring for a hypergraph in an acceptable time. Essentially,

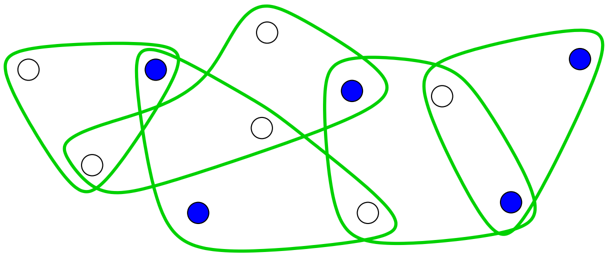

Figure 1.3: A correct 2-coloring for a hypergraph.

the problem that has to be dealt with is that making a hyperedge 2-colored may cause another hyperedge not to be 2-colored any more. Thus, an algorithm for finding a 2-coloring has to provide rules for how to resolve these conflicts.

Example 4: Set Partitioning

Given a system of sets over some set of elements and a collection of c colors, the problem is to partition the elements into c color classes so that the sets are split well. That is, we want to find a coloring such that the maximum number of elements in a set that are of the same color is minimal (such an optimal solution is given in Figure 1.4).

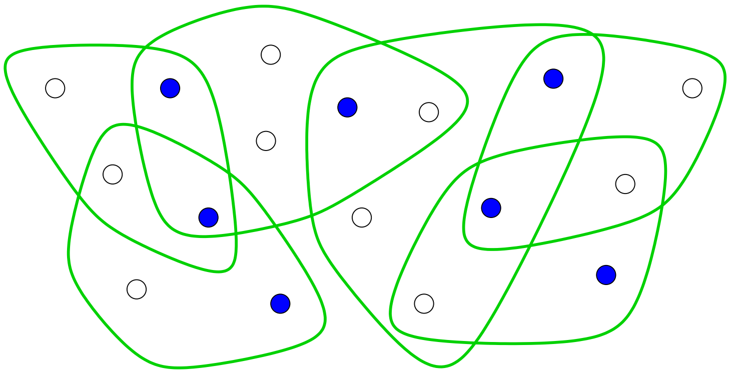

Figure 1.4: An optimal solution to the set partitioning problem for 2 colors.

It is not difficult to imagine that the set partitioning problem involves dealing with even more conflicting interests than in the hypergraph 2-coloring problem.

Example 5: Combinatorial Optimization

Consider the following optimization problem:

A minimax integer program (or MIP in short) has variables $\{x_{i,j} \ : \ i \in [n], \ j \in [\ell_i]\}$, for some integers ℓ_i. Let $N = \sum_{i \in [n]} \ell_i$ and let $\boldsymbol{x}$ denote the N-dimensional vector of the variables $x_{i,j}$. A MIP seeks to minimize a real Y subject to:

(1) a system of linear inequalities $\boldsymbol{Ax} \leq \boldsymbol{y}$, where $\boldsymbol{A} \in [0,1]^{m \times N}$ and $\boldsymbol{y}$ is the m-dimensional vector with the variable Y in each component,

(2) for all $i \in [n]$: $\sum_{j \in [\ell_i]} x_{i,j} = 1$, and

(3) for all i and j: $x_{i,j} \in \{0,1\}$.

MIPs represent a general framework for choice problems (see conditions (2) and (3)). To find an optimal solution Y is NP-hard in general. It requires finding a strategy that can solve the conflicting interests represented by the system of linear inequalities in a best possible way. Thus, we are again searching for good coordination strategies.

We will classify strategies for coordination problems in three different categories.

- *Global versus local*: We characterize an algorithm for a coordination problem as *global* if it deals with the coordination problem as a whole, that is, without splitting it into subproblems that are solved using local objectives. Problems for which global approaches are common are, for instance, the class of P-complete problems. The reason for this may be that, as many believe, P-complete problems cannot be split well into independent subproblems without loosing the ability to find (near-)optimal solutions. Another reason why it may not be possible to split a problem into small subproblems, even if solving the subproblems in an optimal way would give an optimal global solution, could be that the subproblems depend on each other in such a strong way that it is either impossible or extremely difficult to coordinate the selection of solutions for the subproblems in such a way that all of them are solved in a satisfactory way. (Optimizing one subproblem may cause another subproblem to be far from optimal!). However, we will demonstrate in this thesis that for all of the examples given above it is possible (for many of their instances) to split them into subproblems that can be considered (almost) independently.

- *Offline versus online*: An algorithm for a coordination problem is called an *offline* algorithm if it uses or requires us to use the complete information about the problem in order to find a solution. An example for which it is reasonable to assume that complete information is available is the job shop scheduling area. However, in other areas such as the routing area, it is often not reasonable to assume that some central instance (or all nodes of the system) knows the complete problem, since, if true, this would severely limit the scalability of the system. (In the Internet, certainly no one can have a complete knowledge about the routing problem!) In this case, the coordination problem has to be dealt with in an *online* fashion, that is, with only incomplete or sometimes even no information about the problem beforehand. We will demonstrate that for certain problems nevertheless near-optimal solutions can be found in this case.

- *Centralized versus distributed*: We call an algorithm *centralized* if it is designed to run sequentially on a single machine. In contrast to this, an algorithm is called *distributed* if it can be executed in parallel on many different machines. The advantage of centralized algorithms is that they are often much easier to design and verify. However, distributed algorithms have the advantage they may allow the solution of much larger problems. For problems that need a rapid reaction, they are sometimes even the only solution.

In this thesis, our main approach will be to find local algorithms for coordination problems. These algorithms will be either online and distributed, or offline and centralized.

In the last twenty years, the array of techniques for devising and analyzing randomized algorithms has grown rapidly. Often, Hoare's randomized Quicksort algorithm [Hoa62] is cited as one of the first results in this area. Excellent surveys of randomized algorithms were given by Karp [Kar91], Maffioli, Speranza, and Vercellis [MSV85], Welsh [Wel83], and Motwani and

Raghavan [MR95]. In recent years it has turned out that probabilistic techniques are very powerful in solving coordination problems. There are many reasons for this. First of all, randomized algorithms are often much simpler and faster than their deterministic (or derandomized) counterparts. One of the reasons for this is that it is often much easier to avoid bad inputs by using randomized strategies than by using deterministic strategies. Furthermore, randomization often allows us to first solve a hard coordination problem in a relaxed form that allows us to find optimal solutions in an efficient way, and afterwards transform this solution into a (near optimal) solution for the original problem. (Consider, for instance, the widely used randomized rounding technique pioneered by Raghavan and Thompson [RT87].) Moreover, as we will show, randomization can be efficiently used to break a global coordination problem into many small coordination problems that can be treated (almost) independently. This does not only help to speed up the sequential time necessary to find a good solution, but it also enables us to construct efficient distributed algorithms.

Apart from the algorithmic aspect, it has turned out that probabilistic methods can also be used to prove powerful results about the existence of solutions to certain coordination problems. Erdős is usually credited as being the pioneer of the probabilistic method, beginning with his seminal 1947 paper [Erd47] (although the probabilistic method had been used in at least two previous occasions by Turán in 1934 [Tur34] and by Szele in 1943 [Sze43]). Many papers followed (see, for example, [Erd59, Erd61, EL75, AMR91, ES91, LMR94, Spe95, AMM96, Sri96, HMR97, FS98, RS98, Ree98]). Today, probabilistic proof methods are widely recognized as one of the major techniques in the field of combinatorics. An excellent survey of the most important methods in this area can be found in [ASE92]. The basic idea behind the probabilistic method is that in order to prove the existence of a combinatorial object satisfying certain properties (e.g., a graph with neither a large clique nor a large stable set, or a proper coloring of the nodes of a graph) we choose our object at random and prove that with positive probability it satisfies the desired properties.

In this thesis, we will show how to use probabilistic methods to solve various kinds of coordination problems, including the examples given above. We will use these methods in various ways:

- To construct efficient randomized algorithms,
- to analyze the performance of randomized algorithms, and
- to prove upper and lower bounds of existential nature.

For this we will often require the use of tail estimates for sums of independent and dependent random variables and other stochastic techniques. The search for tail estimates was pioneered by Chebychev [Che36] and Markov [Mar13]. In an influential paper, Hoeffding [Hoe63] presented several bounds that enabled the analysis of many randomized algorithms. Other important contributions have, among many others, been due to Chernoff [Che52], Azuma [Azu67], Loéve [Loé77], Nagaev [Nag79], McDiarmid [McD89], Siegel, Schmidt, and Srinivasan [SSS95], and Talagrand [Tal96]. Excellent surveys of tail estimates and probabilistic techniques can be found in [Sie89, GGL95, HMRR98]. To ensure that the thesis is self-contained, we will devote Chapter 3 to their presentation, giving also proofs as long as they are reasonably short.

Chapter 2

Models and Main Results

In this chapter, we introduce some basic notation and the models we will use throughout the thesis. Furthermore, we give a summary of our main results. We start with some basic definitions in graph theory.

2.1 Graph Theory

A *graph* $G = (V, E)$ consists of a set of *nodes* V and a set of *edges* E. The *size* of G is defined as the number of nodes it contains. For all $v, w \in V$, let (v, w) denote a *directed* edge from v to w, and $\{v, w\}$ denote an *undirected* edge connecting v and w. G is called *undirected* if $E \subseteq \{\{v, w\} \mid v, w \in V\}$, and *directed* if $E \subseteq \{(v, w) \mid v, w \in V\}$. Unless explicitly mentioned, we assume in this thesis that G is undirected.

A sequence of contiguous edges in G is called a *path*. The *length* of a path is defined as the number of edges it contains. A path is called *node-simple* if it visits every node in G at most once. Similarly, it is called *edge-simple* (or *simple* in short) if it contains every edge in G at most once. G is called *connected* if, for every pair of nodes $v, w \in V$, there is a path in G from v to w. We call a path a *cycle* if it starts and ends at the same node. G is called a *tree* if it is connected and contains no cycle. A graph $\mathcal{T} = (V', E')$ is called a *spanning tree* of G if $V' = V$, $E' \subseteq E$, and $\mathcal{T}$ is a tree. G is called *bipartite* if its node set can be partitioned into two node sets V_1 and V_2 such that there is no edge connecting two nodes in V_1 or two nodes in V_2.

For any pair of nodes $v, w \in V$, let $\delta(v, w)$ denote the *distance* of v and w in G, that is, the length of a shortest path from v to w. The *diameter* D of G is defined as $\max\{\delta(v, w) : v, w \in V\}$. If $\{v, w\} \in E$ then v is called a *neighbor* of w. The number of neighbors of v is called the *degree* of v and denoted by d_v. The degree of G is defined as $d = \max\{d_v : v \in V\}$. If all nodes in G have the same degree then G is called *regular*.

For directed graphs, we define the number of edges ending at a node v the *in-degree* of v, and the number of edges leaving v the *out-degree* of v.

A family of graphs $\mathcal{G} = \{G_n : n \in \mathbb{N}\}$ has a degree of d if for all $n \in \mathbb{N}$ the degree of G_n is $d(n)$. If it is clear to which family a graph belongs, we say that this graph has a constant (or bounded) degree if and only if its family has a constant degree.

Given a graph $G = (V, E)$, a set $U \subseteq V$ is called an *independent set* if there is no edge in G that has both of its endpoints in U. The following claim implies that it is possible to partition the node set of any graph of degree d into at most $d + 1$ independent sets.

Claim 2.1 *For any graph G of degree d, $d + 1$ colors suffice to color the nodes of G in such a way that no two neighbors in G have the same color.*

Proof. Simply color the nodes one after the other, using one of the $d+1$ given colors in such a way that no two adjacent nodes have the same color. This is always possible, since otherwise there must exist a node in G whose neighbors already use all $d+1$ colors. Because the degree of a node can be at most d, this can not happen. $\square$

2.2 Routing

By "routing" we understand the process of moving information across a network from a source to a destination. Routing involves two basic activities:

- The determination of routing paths, and
- the transport of information groups (typically called *packets*) along the paths.

The first item is usually referred to as *path selection*, and the second item is usually called *packet switching*. (Depending on the research community or research field, both path selection algorithms and packet switching algorithms have been called routing algorithms, but we intend to avoid these ambiguities in this thesis.) In the following, we will only consider routing algorithms in which the path selection must be completed for any packet before sending it to its destination. A popular class of algorithms that often fulfill this property are the so-called *oblivious* algorithms. In the case of oblivious routing, the path of a packet only depends on its source and destination. Before we present our results, we will introduce some notation.

2.2.1 The Basic Model

We consider arbitrary network topologies modeled as undirected graphs of N nodes. The nodes represent switches or processors, and the edges represent bidirectional communication links, unless otherwise stated, with buffers for incoming packets on either side. These buffers are called *edge buffers*, and bounds on the buffer size always refer to the maximum number of packets that these buffers can hold. Additionally, every node contains an *injection buffer*, in which the newly injected packets are initially stored. Routing is performed in a synchronous "store and forward" manner, that is, time proceeds in synchronous steps, and in each step, each edge can be crossed by at most one packet in each direction. Once a packet reaches its destination, it is discarded.

There are two basic classes of routing problems: *static* and *dynamic* routing. In the case of static routing it is assumed that all packets are injected at the same time, whereas in the case of dynamic routing it is assumed that new packets are continuously injected into the network.

Static packet routing

In the static packet routing model, we are given a collection of packets that are specified by source–destination pairs. Initially, all packets are in the injection buffers of their source. The goal is to send the packets along the edges of the network to their destinations in such a way that the *routing time*, i.e., the time needed to deliver all packets, is minimized. (The routing time is usually measured in the number of communication steps, neglecting the time required for internal computations.) To achieve this goal, algorithms are sought that allow the selection of good paths for the packets and the efficient coordination of the packet movements along these paths. We distinguish between two kinds of packet routing strategies, often referred to as *routing protocols*: *Offline* protocols and *online* protocols.

In the case of offline protocols, we allow all processors of a parallel system to have a complete knowledge about the routing problem. This enables them, in principle, to develop an optimal strategy for sending the packets to their destinations (recall that we do not count internal computations, but only communication steps). In this area, important questions are, how fast such a protocol can be, and whether protocols with an (asymptotically) optimal routing time can be computed in polynomial time.

In the case of online protocols, at the beginning every processor only knows the packets that start in it (and, maybe, a few global parameters of the routing problem). Therefore, decisions can only be based on the routing information carried by the packets and on the local history of the execution.

Offline and online routing protocols (i.e., protocols that solve both the path selection and the switching problem) have rarely been presented so far. Often it is assumed that a fixed collection of paths is already given along which the packets have to be sent. In this case, it remains to solve a *static packet switching* problem. The most important parameters that have been used in the literature to bound the runtime of protocols for the static packet switching problem are the *congestion* C and *dilation* D. C denotes the maximum number of paths crossing an edge in the same direction, and D denotes the maximum length of a path. The goal is to send the packets along the given paths in such a way that the routing time is minimized. Similar to above, we distinguish between two kinds of packet switching strategies, often referred to as *switching protocols*: *Offline* protocols and *online* protocols. As we will see, it is possible to construct simple, efficient online switching protocols for very general classes of path collections.

Dynamic packet routing

In the dynamic packet routing model, we assume that packets, specified by source-destination pairs, are continuously injected into the system. Important goals in this area are to select paths and strategies for the coordination of the packet movements along the paths such that the number of packets in the system does not grow unboundedly with time and the packet delays are low.

As in the static packet routing model, strategies have rarely been presented for the dynamic packet routing model. Usually, it is assumed that for each injected packet a path is already specified along which the packet has to be sent. This model is called the *dynamic packet switching* model. Within this model, the most commonly used injection models are the *stochastic* and the *adversarial* injection model.

In the stochastic model, the packets are injected by a set of *generators*, each of them mapped to one of the nodes in the network. We allow any relationship between the number of generators and the number of nodes in the network. Furthermore, we place no restrictions on the distribution of the generators among the nodes. That is, one node could have several generators, whereas another node may have none. So a generator may represent a user thread or process, whereas a node may represent a processor. In each time step, each generator g placed on a node v injects a packet with some fixed probability p_g. This probability is called the *injection rate* of g. For each packet, the generator randomly selects a destination and a routing path from v to this destination according to an arbitrary, fixed probability distribution. We assume that each generator is operating independently of other generators, and the injection of a packet and its routing path is independent of injections in previous time steps. However, we do not demand that the destinations are chosen uniformly from the set of all nodes, or that packets with the same source and destination node must follow the same routing path. Let λ_e be the expected number of packets generated in a time step that contain the edge e in their routing paths. Then the *(overall) injection rate* λ is defined as the maximum λ_e over all edges.

In the adversarial model, an adversary is allowed to demand network bandwidth up to a prescribed injection rate. For any $w, \lambda > 0$, an adversary is called a *bounded adversary of rate* (w, λ) if for all edges e and all time intervals I of length w, it injects no more than $\lambda \cdot w$ packets during I that contain edge e in their routing paths. As in the stochastic model, λ is defined to be the *injection rate*. (We use the adversarial model as defined by Andrews et al. [AAF$^+$96] rather than the original model introduced by Borodin et al. [BKR$^+$96], because the former model avoids calculating with floors and ceilings. Apart from minor changes in constants, however, all of the results presented in this thesis also hold in the original model of Borodin et al..)

For both injection models, a protocol is called *stable* for a given injection rate λ if the (expected) number of packets in the injection and edge buffers does not grow to infinity with time. We are interested in protocols that are stable for high injection rates. Of course, since an edge can transport at most one packet per step, λ can be at most 1. Our aim is to construct algorithms that are stable under injection rates that are close to 1. Additionally, we are interested in short delays for the packets, i.e., we aim to minimize the time from injection to service for each packet.

Apart from the stability and the routing time, we will consider another property of dynamic switching protocols, the recovery from worst case scenarios. Although a switching protocol may ensure that bad configurations are very unlikely, they may eventually occur from time to time when the switching protocol runs for an infinite number of time steps. Let a *worst case scenario* denote an arbitrarily bad configuration of the network. Then the *recovery time* with regard to some property $\mathcal{P}$ of the switching protocol is defined as the time that has to pass by after the occurrence of a worst case scenario until $\mathcal{P}$ holds again. (In our case, we are interested in properties such as the expected routing time of a packet and time bounds that hold with high probability.)

2.2.2 Main Results

In Chapters 4 to 6, we consider three different problems in the area of routing theory:

- transforming static switching protocols into dynamic switching protocols [SV99],
- constructing switching protocols for which the routing time of a packet only depends on local parameters [BS99], and
- routing in ad hoc wireless networks [AS98].

In the following, we give a brief summary of these results and discuss their historical background.

Transforming static into dynamic protocols

Many static online switching protocols have been developed in recent years (see, e.g., [LMR94, LMRR94, MV95, RT96, CMSV96, OR97]). In practice however, networks are rarely used in this static fashion but packets are injected dynamically into the network. Since much less is known in the area of dynamic switching than in the area of static switching, it has been considered as highly desirable to transfer the results gathered for the static switching model to the dynamic switching model. Previous to our research, it was known only for very restricted classes of static online protocols that they also yield good dynamic protocols (see, e.g., [AAF$^+$96, BFU96, RT96, SV96]). We found a black box method presented in [SV99] that allows us to transform *any* static online switching protocol into an efficient dynamic online switching protocol for both the stochastic and the adversarial injection model. Our black box transformation ensures, for instance, that if the static algorithm is pure (i.e., no extra packets apart from the original packets are sent), its dynamic variant is stable up to a maximum possible injection rate. Furthermore, in the stochastic model, the routing time of a packet only depends on local parameters such as the length of its routing path, rather than on the maximum possible path length, even if the static algorithm chosen for the transformation does not provide this locality feature and is not pure. In the latter case, however, the dynamic variant may not be stable any more up to a maximum possible injection rate. In the adversarial model, the delay bound of the packets is closely related to the time bound given for the static algorithm. Moreover, we proved that for any static switching protocol, its dynamic variant recovers quickly from any worst case scenario (concerning, for instance, the expected delay of a packet).

Locally efficient switching protocols

Most of the work done in the area of static switching concentrates on minimizing the runtime of the whole schedule rather than minimizing the runtime of individual packets, using global parameters such as the congestion and dilation of a path collection. Asymptotically (almost) optimal online protocols for this model can be found, for instance, in [LMRR94, MV95, CMSV96, RT96, OR97]). We studied in [BS99] the problem of minimizing the routing time of individual packets, using local parameters:

Assume that a fixed collection of paths is given. For every path p in this collection, let c_p denote the maximum, over all edges e in p, of the number of paths crossing e, and let d_p denote the length of p. Find a schedule (that does *not* require to know c_p and d_p) such that the routing time of a packet following a path p merely depends on c_p and d_p.

We were the first who found efficient online protocols for this model. In fact, it is shown in [BS99] that the routing time of any packet can be bounded by $O(c_p + d_p \log\log n)$, with high probability. We also showed how to use our static protocols to obtain dynamic switching protocols that are stable under very general assumptions that cover all stochastic and adversarial injection models known to us.

Routing in ad hoc wireless networks

An *ad hoc* wireless network is a collection of wireless mobile hosts forming a temporary network without the aid of any established infrastructure or centralized administration. This type of network is of great importance in situations where it is very difficult to provide the necessary infrastructure, but it is a challenging task to enable a fast and reliable communication within such a network. Previous to our work, people mainly looked at finding efficient routing strategies for ad hoc networks formed by mobile stations that have a fixed transmission power. Many upper and lower bounds have already been shown for this model (see, e.g., [ABLP91, KM93, BGI92, Hwa95, BP97] for lower bounds, and [CK85, Wei87, CW91, BGI92, GM95, KR95] for upper bounds).

In [AS98], we instead modeled and analyzed the performance of so-called *power-controlled* ad hoc wireless networks: networks where the mobile stations are able to change their transmission power. We concentrated on finding schemes for routing arbitrary permutations in these networks. In general, it is NP-hard even to find a $n^{1-\epsilon}$-approximation for any constant ϵ to the fastest possible strategy for routing a given permutation problem on n mobile stations. However, we demonstrated that if we allow ourselves to consider slightly less general problems, efficient solutions can be found.

First, we showed in [AS98] that there is a natural class of distributed schemes for handling node-to-node communication on top of which online path selection and switching strategies can be constructed such that the performance of this class of schemes can be exploited in a nearly optimal way for routing permutations in any static power-controlled ad hoc network. We then demonstrated that if we restrict ourselves to the important case of routing between nodes distributed randomly in a Euclidean space, we can route in a time that is asymptotically optimal for any routing scheme.

2.3 Job Shop Scheduling

In Chapter 7, we consider the problem of finding solutions to job shop scheduling problems. Our results are based on [FS98]. First, we introduce the basic model.

2.3.1 The Basic Model

In job shop scheduling problems we are given n jobs and m machines. Each job is composed of a sequence of operations, where each operation has a machine and a *length* (processing time) associated with it. An operation is scheduled by specifying a time step at which it begins. From this time point the operation has to executed on its respective machine without interruption for a time period that is equal to its length. A *legal* schedule is one in which each operation is scheduled only after all operations preceding it in its job have been completed, and each machine performs at most one operation at any unit of time. The *makespan* of a legal schedule is the time by which all jobs have been completed. For a given job shop scheduling instance, let L denote the minimum makespan, taken over all possible legal schedules. For the special case of *acyclic* job shop scheduling, we require that all operations of a job have to be processed on different machines.

2.3.2 Main Results

There are two trivial lower bounds on L. One of them is the length of the longest job, denoted here as the dilation D. The other is the load on the most loaded machine, i.e., the total number of time units that all jobs require on this machine, which is called here the congestion C. Obviously, $lb = \max[C, D]$ is a lower bound on L. A celebrated result of Leighton, Maggs, and Rao [LMR88] shows that if all operations are of the same length and no job has more than one operation at any machine, then there always is a legal schedule of length $O(lb)$, independent of n and m. For the general case, Shmoys, Stein, and Wein [SSW94] showed that there always is a legal schedule of length $\tilde{O}(lb(\log lb)^2)$, where the $(\tilde{O})$ notation is used to suppress $\log\log(lb)$ terms. There was no better lower bound known than $\Omega(lb)$.

In [FS98] we looked at the problem of finding upper and lower bounds for the acyclic job shop scheduling problem (no job has more than one operation at any machine). For this class of job shop scheduling problems, we improved the upper bound to $\tilde{O}(lb \log lb)$. We also showed that our new upper bound is essentially best possible, by proving the existence of instances of acyclic job shop scheduling for which the shortest legal schedule is of length $\tilde{\Omega}(lb \log lb)$. This resolved (negatively) the question of whether the linear upper bound of Leighton, Maggs, and Rao applies to arbitrary job shop scheduling instances (without the restriction to acyclicity and operations of equal length). Consequences of our results include nearly tight upper and lower bounds for the problem of sending packets of uniform size in arbitrary non-uniform networks and sending non-uniform packets in uniform networks.

2.4 Algorithmic Forms of the Lovász Local Lemma

Probabilistic methods have been used with great success to prove the existence of objects with desirable properties by showing that a randomly chosen object from an appropriate probability distribution has the desired properties with positive probability. In most applications this probability is not only positive, but is actually high and frequently tends to 1 as the parameters of the problem tend to infinity. In such cases, the proof usually supplies an efficient randomized algorithm for producing a structure of the desired type.

There are, however, certain examples, where one can prove the existence of the required combinatorial structure by probabilistic arguments that deal with rare events; events that hold with positive probability which is exponentially small in the size of the input. This happens often when using the Lovász Local Lemma [EL75]. We present this lemma in its general form together with some notation.

2.4.1 Basic Notation

A directed graph $G = (V, E)$ on the set of nodes $V = \{1, 2, \ldots, n\}$ is called a *dependency graph* for the events $A_1, \ldots, A_n$ if for every i, $1 \leq i \leq n$, the event A_i is independent of any set of events A_j with $(i, j) \notin E$.

Lemma 2.2 (Lovász Local Lemma) *Let $A_1, \ldots, A_n$ be "bad" events in an arbitrary probability space. Suppose that $G = (V, E)$ is a dependency graph of these events and suppose there are real numbers $x_i \in [0, 1)$ for all $1 \leq i \leq n$ with*

$$\Pr[A_i] \leq x_i \prod_{(i,j)\in E} (1 - x_j)$$

for all $1 \leq i \leq n$. Then,

$$\Pr\left[\bigcap_{i=1}^{n} \bar{A}_i\right] \geq \prod_{i=1}^{n}(1 - x_i) \ .$$

In particular, with positive probability no bad event A_i holds.

In this thesis, we will demonstrate how to construct algorithmic forms of this lemma, concentrating on the following two problems.

Hypergraph coloring

A *hypergraph* $H = (V, E)$ consists of a set of nodes V and a set of *hyperedges* $E \subseteq \mathcal{P}(V)$. H is called *uniform* if all hyperedges are of the same cardinality.

A hypergraph $H = (V, E)$ is *c-colorable* if c colors suffice to color its nodes in such a way that no hyperedge in E is monochromatic (i.e., has only nodes of a single color).

Minimax integer programs

A minimax integer program (or MIP in short) has variables $\{x_{i,j} \; : \; i \in [n], \; j \in [\ell_i]\}$, for some integers ℓ_i. Let $N = \sum_{i \in [n]} \ell_i$ and let $\boldsymbol{x}$ denote the N-dimensional vector of the variables $x_{i,j}$. A MIP seeks to minimize a real Y subject to:

(1) a system of linear inequalities $\boldsymbol{Ax} \leq \boldsymbol{y}$, where $\boldsymbol{A} \in [0,1]^{m \times N}$ and $\boldsymbol{y}$ is the m-dimensional vector with the variable Y in each component,
(2) for all $i \in [n]$: $\sum_{j \in [\ell_i]} x_{i,j} = 1$, and
(3) for all i and j: $x_{i,j} \in \{0,1\}$.

2.4.2 Main Results

In Chapters 8 and 9, we consider the problem of

- coloring hypergraphs [CS00a] and
- finding good approximate solutions to MIPs [CS00b].

In the following, we give a brief summary and present the historical background of our results.

Coloring non-uniform hypergraphs

To turn proofs using the Lovász Local Lemma into efficient algorithms, even random ones, proved to be difficult for many of its applications. For the hypergraph 2-coloring problem, for instance, strong existential results are already known since 1975 [EL75]. However, it was not before 1991 that a polynomial time algorithm was found for some of these hypergraphs. This was achieved in a breakthrough paper by Beck [Bec91]. Alon [Alo91] provided a parallel variant of the algorithm and simplified the arguments used. His method was further generalized by Molloy and Reed [MR98] to yield efficient algorithms for a number of applications of the Lovász Local Lemma. However, all of these approaches are limited to hypergraphs that only require a simplified, symmetric form of the Lovász Local Lemma, in which a very regular structure of the events under consideration is required. In [CS00a] we were the first to present a polynomial time algorithm that also finds 2-colorings for hypergraphs that require the Lovász Local Lemma in its general form. Under certain circumstances, our algorithm even runs in expected linear time. Even for those hypergraphs that only require a simplified form of the Lovász Local Lemma such a runtime bound was not known before.

Finding approximate solutions to MIPs

MIPs represent a general framework for choice problems (see properties (2) and (3)). Many applications in the area of routing and scheduling can be formulated as MIPs (for several examples, see [LRS99]). A common strategy to find good approximate solutions to MIPs is to first solve its linear relaxation and then to use randomized rounding. This approach was pioneered by Raghavan and Thompson [RT87] and was later extended and refined for MIPs by Srinivasan [Sri96], Lu [Lu99], and Leighton, Rao, and Srinivasan [LRS99]. The latter papers apply a simplified form of the Lovász Local Lemma to construct good rounding strategies. We found polynomial time algorithms also for MIPs that require the use of the general Lovász Local Lemma in order to find good approximate solutions. The great advantage of our result compared to the results in [LRS99, Lu99] is that local dependencies among the rows of a MIP can be exploited, whereas in [LRS99, Lu99] only the maximum dependency of a row on other rows can be used for the rounding. As we demonstrated, the property of being able to exploit local dependencies is vital to constructivize the job shop scheduling results presented in [FS98]. We also showed how to apply our method to obtain the first approximation algorithm for MAX SAT problems that is able to exploit local dependencies.

2.5 Probabilistic Methods

In order to analyze our algorithms and existential results presented in this thesis, we require a variety of probabilistic techniques. An introduction to these techniques is given in the next chapter. Most of the results presented there have been collected from various books and research reports in the area of probability theory and probabilistic methods. Techniques based on own research are, for instance, a Chernoff bound for sums of geometrically distributed random variables (Theorem 3.38), a tail estimate for sums of random variables whose dependencies are expressed via dependency graphs (Theorem 3.50), and a tail estimate for non-uniformly dependent random variables (Theorem 3.52). Moreover, some parts of Section 3.5 and those parts in Section 3.6.2 that present extensions to Alon's and Molloy and Reed's method are new.

Chapter 3

Introduction to Probability Theory

In this chapter, we review the basic concepts in the area of probability theory and present probabilistic techniques that allow the solution of a wide range of coordination problems. A special emphasis will be given on modeling dependence and presenting different tools to deal with dependent events or dependent random variables. Our intention will not be to give a mathematically rigorous introduction with all proofs, but rather to concentrate on the main ideas and techniques that lead to the results presented in this section.

3.1 Basic Notation and Concepts

In the following, small letters usually represent scalars, capital letters represent sets or events, and letters in bold face represent vectors (like $\boldsymbol{a}, \boldsymbol{b}, \ldots$) or matrices (like $\boldsymbol{A}, \boldsymbol{B}, \ldots$). A family of sets is denoted by $\mathcal{A}, \mathcal{B}, \ldots$.

3.1.1 Set Theory

Let M be an arbitrary set. Given any subset $A \subseteq M$, $|A|$ denotes its *cardinality*, that is, the number of elements in A, and $\bar{A}$ denotes its *complement* $M \setminus A$. $\mathcal{P}(M)$ is called the *power set* of M and consists of all subsets $A \subseteq M$. We summarize a few standard sets.

- $\emptyset$ denotes the empty set.
- $\mathbb{N}$ means the set of integers $\{1, 2, 3, 4, \ldots\}$, and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.
- For any $c \in \mathbb{N}$, the set $[c]$ represents the set $\{1, \ldots, c\}$.
- $\mathbb{R}_+$ denotes the set of all non-negative reals.
- For any two reals $x, y \in \mathbb{R}$ with $x < y$, $[x, y]$ denotes the set of all real numbers z with $x \leq z \leq y$, and (x, y) denotes the set of all real numbers z with $x < z < y$. (Also mixed forms $[x, y)$ are possible.)
- For any $n \in \mathbb{N}$, S_n denotes the set of all permutations on $[n]$.

Given two sets A and B,

- $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$ is called the *intersection* of A and B,
- $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$ is called the *union* of A and B, and

- $A \setminus B = \{x \mid x \in A \text{ and } x \notin B\}$ is called the *difference* of A and B.

Two sets A and B are called *disjoint* if $A \cap B = \emptyset$.

A very important principle in set theory is the *inclusion-exclusion principle*: Consider any collection of n sets $A_1, \ldots, A_n$. Then,

$$\left| \bigcup_{i=1}^{n} A_i \right| = \sum_{k=1}^{n} (-1)^{k+1} \sum_{i_1 < i_2 < \ldots < i_k} \left| \bigcap_{j=1}^{k} A_{i_j} \right| .$$

3.1.2 Combinatorics

Basic Notation

We frequently use the following notation.

- If we use "e" in formulas, we always mean the Euler number (2.71828..).
- By "log" we mean the logarithm to the base of 2. "ln" denotes the logarithm to the base of e. If we use any other base, say b, we write "$\log_b$".
- $\log^k n$ means $(\log n)^k$.
- $n! = n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1$ is called *n factorial*. By definition, $0! = 1$. For any $x \in \mathbb{R}$, the *rth descending factorial* of x is $x^{(r)} = x \cdot (x-1) \cdot \ldots \cdot (x-r+1)$, and the *$r$th ascending factorial* of x is $x^{[r]} = x \cdot (x+1) \cdot \ldots \cdot (x+r-1)$. By definition, $x^{[0]} = x^{(0)} = 1$ for every $x \in \mathbb{R}$.
- For all $0 \le k \le n$,
$$\binom{n}{k} = \frac{n!}{(n-k)! \cdot k!} = \frac{n^{(k)}}{k!} .$$

Identities

We will require sometimes to switch between different representations. For this, the following list of identities will be helpful. For every $x \in \mathbb{R}$,

$$\mathrm{e}^x = \sum_{i=0}^{\infty} \frac{x^i}{i!} \quad \text{and} \quad \ln(1+x) = \sum_{i=1}^{\infty} \frac{(-1)^{i-1}}{i} \cdot x^i .$$

For every $n \ge 0$ and $x \in \mathbb{R}$ it holds that

$$x^{(n)} = \sum_{k=0}^{n} s_{n,k} \cdot x^k$$

where the $s_{n,k}$'s are called the *Stirling numbers of the first kind*. They are defined as $s_{n,k} = (-1)^{n-k} c_{n,k}$ for all $k, n \in \mathbb{N}_0$, where $c_{n,k}$ is called a *signless* Stirling number of the first kind. The $c_{n,k}$'s are recursively defined by

$$\begin{aligned} c_{0,0} &= 1 \\ c_{n,0} &= 0 \quad \text{for all } n \in \mathbb{N} \\ c_{n,k} &= (n-1) \cdot c_{n-1,k} + c_{n-1,k-1} \quad \text{for all } k, n \in \mathbb{N}. \end{aligned}$$

n/k	0	1	2	3	4	5	6	7	8
0	1								
1		1							
2		1	1						
3		2	3	1					
4		6	11	6	1				
5		24	50	35	10	1			
6		120	274	225	85	15	1		
7		720	1764	1624	735	175	21	1	
8		5040	1368	13132	6769	1960	322	28	1

Figure 3.1: Table of the signless Stirling numbers of the first kind

$c_{n,k}$ represents the number of permutations $\pi \in S_n$ with exactly k cycles. Some values of the $c_{n,k}$'s are given in Figure 3.1.

Furthermore, for every $n \geq 0$ and $x \in \mathrm{I\!R}$ it holds that

$$x^n = \sum_{k=0}^{n} S_{n,k} \cdot x^{(k)}$$

where the $S_{n,k}$'s are called *Stirling numbers of the second kind*. They are recursively defined by

$$\begin{aligned} S_{n,k} &= 0 \quad \text{for } k = 0 \text{ or } k > n \\ S_{n,1} &= S_{n,n} = 1 \\ S_{n,k} &= k \cdot S_{n-1,k} + S_{n-1,k-1} \quad \text{for all } k, n \in \mathrm{I\!N} \end{aligned}$$

$S_{n,k}$ represents the number of ways of partitioning $[n]$ into k non-empty sets. Some values of the $S_{n,k}$'s are given in Figure 3.2.

n/k	0	1	2	3	4	5	6	7	8
0	1								
1		1							
2		1	1						
3		1	3	1					
4		1	7	6	1				
5		1	15	25	10	1			
6		1	31	90	65	15	1		
7		1	63	301	350	140	21	1	
8		1	127	966	1701	1050	266	28	1

Figure 3.2: Table of the Stirling numbers of the second kind

Furthermore, for every $n \geq 0$ and $x \in \mathrm{I\!R}$ we have

$$x^{[n]} = \sum_{k=0}^{n} c_{n,k} \cdot x^k$$

and

$$x^n = \sum_{k=0}^{n} (-1)^{n-k} S_{n,k} \cdot x^{[k]} .$$

Useful inequalities

Using the identity for e^x above, one can show that

$$1 + x \leq \mathrm{e}^x \quad \text{for all } x \in \mathbb{R}$$

and

$$1 - x \leq \mathrm{e}^{-x} \leq 1 - x + x^2/2 \quad \text{for all } x \in [0, 1].$$

Furthermore,

$$\left(1 + \frac{1}{n}\right)^n \leq \mathrm{e} \leq \left(1 + \frac{1}{n}\right)^{n+1}$$

for all $n \in \mathbb{N}$, implying that $\lim_{n\to\infty}(1 + 1/n)^n = \mathrm{e}$. Also,

$$\left(1 - \frac{1}{n}\right)^n \leq \frac{1}{\mathrm{e}} \leq \left(1 - \frac{1}{n}\right)^{n-1}$$

for all $n \in \mathbb{N}$, implying that $\lim_{n\to\infty}(1 - 1/n)^n = 1/\mathrm{e}$.

Frequently, one wishes to express $n!$ or $\binom{n}{k}$ by terms based on standard operations. For all $n \in \mathbb{N}$, Stirling's formula says that

$$n! = \alpha(n)\sqrt{2\pi n}\left(\frac{n}{\mathrm{e}}\right)^n \quad \text{where } \alpha(n) \in [1, \mathrm{e}^{1/12n}] \; .$$

Often, the simpler inequality $(n/\mathrm{e})^n \leq n! \leq n^n$ is used.

Stirling's formula implies that for all $k, n \in \mathbb{N}$ with $k < n$,

$$\binom{n}{k} = \beta(n,k)\left(\frac{n}{k}\right)^k \left(\frac{n}{n-k}\right)^{n-k} \sqrt{\frac{n}{2\pi k(n-k)}} \quad \text{where } \beta(n,k) \in [\mathrm{e}^{-1/6k}, 1] \; .$$

Often, the simpler inequality $(n/k)^k \leq \binom{n}{k} \leq (\mathrm{e}n/k)^k$ is used.

For any $0 < \alpha \leq 1/2$,

$$\sum_{0 \leq k \leq \alpha n} \binom{n}{k} \leq 2^{H(\alpha)\cdot n} \; ,$$

where $H(\alpha) = -\alpha \log \alpha - (1-\alpha)\log(1-\alpha)$ is the binary entropy function.

We list some other important inequalities taken from [Hof87]. For any $n \in \mathbb{N}$, let $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ be any two sequences of non-negative real numbers.

Fact 3.1 (Chebychev) *If $a_1 \geq a_2 \geq \ldots \geq a_n$ and $b_1 \geq b_2 \geq \ldots \geq b_n$ then*

$$n\sum_{i=1}^n a_i b_i \geq \left(\sum_{i=1}^n a_i\right)\left(\sum_{i=1}^n b_i\right) \; .$$

Fact 3.2 (Hölder) *If $p > 1$, $q > 1$ and $p^{-1} + q^{-1} = 1$, then*

$$\sum_{i=1}^n a_i b_i \leq \left(\sum_{i=1}^n a_i^p\right)^{1/p}\left(\sum_{i=1}^n b_i^q\right)^{1/q} \; .$$

For the special case of $p = q = 2$, this inequality is called the Cauchy-Schwartz inequality.

Fact 3.3 (Minkowski) *If $p > 1$ then*

$$\left(\sum_{i=1}^{n}(a_i + b_i)^p\right)^{1/p} \leq \left(\sum_{i=1}^{n} a_i^p\right)^{1/p} + \left(\sum_{i=1}^{n} b_i^p\right)^{1/p} .$$

Fact 3.4 (The Three Means) *Given a sequence of real numbers $x_1, \ldots, x_n$,*

- *its* arithmetic mean *is defined as $A = \sum_i x_i/n$,*
- *its* geometric mean *is defined as $G = (\prod_i x_i)^{1/n}$, and*
- *its* harmonic mean *is defined as $H = (\sum_i x_i^{-1}/n)^{-1}$.*

If $x_1, \ldots, x_n$ are non-negative, then

$$A \geq G \geq H .$$

Basic Counting

Many combinatorial problems can be viewed as drawing balls. Consider the problem of determining the number of ways of drawing k balls one after the other out of n balls, numbered from 1 to n. The most important cases are summarized in Figure 3.3.

	ordered	unordered
with replacement	n^k	$\binom{n+k-1}{k}$
without replacement	$n^{(k)}$	$\binom{n}{k}$

Figure 3.3: Number of outcomes of drawing k out of n balls

3.2 Basic Concepts in Probability Theory

In this section we will introduce some basic concepts in probability theory.

Let us consider a random experiment of which all possible results are included in a non-empty set Ω, usually called the *sample space*. An element $\omega \in \Omega$ is called a *sample point* or *outcome* of the experiment. An *event* of a random experiment is specified as a subset of Ω. Event A is called *true* if an outcome $\omega \in \Omega$ has been chosen with $\omega \in A$. Otherwise A is called *false*. A system $\mathcal{A}$ of events (or, in general, subsets of Ω) is called an *algebra* if

- $\Omega \in \mathcal{A}$,
- if $A, B \in \mathcal{A}$, then $A \cup B \in \mathcal{A}$ and $A \cap B \in \mathcal{A}$, and
- if $A \in \mathcal{A}$, then $\bar{A} \in \mathcal{A}$.

Given an algebra $\mathcal{A}$, a function $\mu : \mathcal{A} \to \mathbb{R}_+$ is called a *measure* on $\mathcal{A}$ if for every pair of disjoint sets $A, B \in \mathcal{A}$ we have

$$\mu(A \cup B) = \mu(A) + \mu(B) .$$

This definition clearly implies that $\mu(\emptyset) = 0$ and that for any set of pairwise disjoint events $A_1, \ldots, A_k \in \mathcal{A}$ we have

$$\mu\left(\bigcup_{i=1}^{k} A_i\right) = \sum_{i=1}^{n} \mu(A_i) .$$

Furthermore, it implies that for any pair of sets $A, B \in \mathcal{A}$ we have

$$\mu(A \cup B) = \mu(A) + \mu(B) - \mu(A \cap B) .$$

We say that a function $p : \mathcal{A} \to [0, 1]$ is a *probability measure* if

- p is a measure on $\mathcal{A}$ and
- $p(\Omega) = 1$.

Given a probability measure p, the *probability* of an event A to be true is defined as

$$\Pr[A] = p(A) .$$

We say that a triple $(\Omega, \mathcal{A}, p)$ is a *probability space* if $\mathcal{A}$ is an algebra over Ω and p is a probability measure on $\mathcal{A}$.

3.2.1 Events

Starting from a given collection of sets that represent events, we can form new events by means of statements containing the logical connectives "or," "and," and "not," which correspond in the language of set theory to the operations "union," "intersection," and "complement."

If A and B are events, their *union*, denoted by $A \cup B$, is the event consisting of all outcomes realizing either A or B. The *intersection* of A and B, denoted by $A \cap B$, consists of all outcomes realizing both A and B. The *difference* of B and A, denoted by $B \setminus A$, consists of all outcomes that belong to B but not to A. If A is a subset of Ω, its *complement*, denoted by $\bar{A}$, is the set of outcomes in Ω that do not belong to A. That is, $\bar{A} = \Omega \setminus A$.

Two events A and B are called *disjoint* if $A \cap B$ is empty. In probability theory, $\emptyset$ is called the *impossible* event. The set Ω is naturally called the *certain* event.

If Ω is a countable sample space (i.e., its elements can be arranged in a sequence so that the rth element is identifiable for any $r \in \mathbb{N}$), we define the *size* of an event A, denoted by $|A|$, to be the number of outcomes it contains.

3.2.2 The Inclusion-Exclusion Principle

Let $A_1, \ldots, A_n$ be any collection of events. The *inclusion-exclusion principle* stated in Section 3.1.1 implies that

$$\Pr\left[\bigcup_{i=1}^{n} A_i\right] = \sum_{k=1}^{n} (-1)^{k+1} \sum_{i_1 < i_2 < \ldots < i_k} \Pr\left[\bigcap_{j=1}^{k} A_{i_j}\right] . \tag{3.1}$$

For the special case of $n = 2$ we obtain

$$\Pr[A_1 \cup A_2] = \Pr[A_1] + \Pr[A_2] - \Pr[A_1 \cap A_2]\ .$$

In cases where it is too difficult to evaluate (3.1) exactly, *Bonferroni's inequalities* may be used to find suitable approximations:

- For every odd m,

$$\Pr\left[\bigcup_{i=1}^{n} A_i\right] \leq \sum_{k=1}^{m} (-1)^{k+1} \sum_{i_1 < i_2 < \ldots < i_k} \Pr\left[\bigcap_{j=1}^{k} A_{i_j}\right]\ .$$

- For every even m,

$$\Pr\left[\bigcup_{i=1}^{n} A_i\right] \geq \sum_{k=1}^{m} (-1)^{k+1} \sum_{i_1 < i_2 < \ldots < i_k} \Pr\left[\bigcap_{j=1}^{k} A_{i_j}\right]\ .$$

Special cases of these inequalities are *Boole's inequalities*:

$$\Pr\left[\bigcup_{i=1}^{n} A_i\right] \leq \sum_{i=1}^{n} \Pr[A_i]$$

and

$$\Pr\left[\bigcup_{i=1}^{n} A_i\right] \geq \sum_{i=1}^{n} \Pr[A_i] - \sum_{1 \leq i < j \leq n} \Pr[A_i \cap A_j]\ .$$

We demonstrate the usefulness of the inclusion-exclusion principle by some simple examples.

Example 1: Consider the problem of coloring the nodes of a hypergraph G with the help of 2 colors such that no hyperedge is monochromatic (i.e., no hyperedge only contains nodes of a single color). We would like to identify a class of hypergraphs for which this is always possible. Such a class is given in the following claim.

Claim 3.5 *For every hypergraph G with n hyperedges in which every hyperedge is of size at least $\log n + 2$, there is a 2-coloring of the nodes such that no hyperedge is monochromatic.*

Proof. Consider the random experiment of choosing for each node independently and uniformly at random one of the two possible colors. In this case, the probability that a hyperedge of size k is monochromatic is equal to $2 \cdot 2^{-k} = 2^{-k+1}$. Assume the hyperedges to be numbered from 1 to n and let A_i be the event that hyperedge i is monochromatic. Then, by the inclusion-exclusion principle,

$$\Pr[A_1 \cup \ldots \cup A_n] \leq \sum_{i=1}^{n} \Pr[A_i] \leq \sum_{i=1}^{n} 2^{-(\log n + 1)} = \frac{1}{2}\ .$$

Hence,

$$\Pr[\bar{A}_1 \cap \ldots \cap \bar{A}_n] = 1 - \Pr[A_1 \cup \ldots \cup A_n] \geq \frac{1}{2}\ ,$$

and therefore there must exist a 2-coloring such that no edge is monochromatic. □

Example 2: Assume that we have n balls and n bins. Each ball is placed in a bin that is chosen independently and uniformly at random. The goal is to provide an upper bound for the maximum number of balls in a bin. This will be given in the following claim.

Claim 3.6 *With a probability of at least* $1/2$*, the maximum number of balls in a bin is at most* $(1 + o(1)) \log n / \log \log n$*.*

Proof. Let $k = \alpha \log n / \log \log n$ for some non-negative α that will be specified later. For every $i \in [n]$, let the event A_i be true if and only if bin i has at least k balls. Furthermore, let E_1 denote the event that there exists a bin with at least k balls. Then, by Boole's inequality,

$$\Pr[E_1] = \Pr[A_1 \cup \ldots \cup A_n] \leq \sum_{i=1}^{n} \Pr[A_i] \ .$$

For any subset $S \subseteq [n]$, let the event $A_{i,S}$ be true if and only if all balls in S are placed in bin i. Then, again by Boole's inequality,

$$\Pr[A_i] = \Pr\left[\bigcup_{S \subseteq [n],\ |S|=k} A_{i,S}\right] \leq \sum_{S \subseteq [n],\ |S|=k} \Pr[A_{i,S}]$$

for all i. Since $|\{S \subseteq [n],\ |S| = k\}| = \binom{n}{k}$ and $\Pr[A_{i,S}] = 1/n^{|S|}$ for all $S \subseteq [n]$, we obtain

$$\begin{aligned}
\Pr[E_1] &\leq n \cdot \binom{n}{k} \left(\frac{1}{n}\right)^k \\
&\leq n \cdot \left(\frac{\mathrm{e} \cdot n}{k}\right)^k \left(\frac{1}{n}\right)^k = n \cdot \left(\frac{\mathrm{e}}{k}\right)^k \\
&\leq n \cdot 2^{-\alpha \frac{\log n}{\log \log n} \cdot \log\left(\frac{\alpha \log n}{\mathrm{e} \log \log n}\right)} \\
&\leq n \cdot 2^{-\log n - 1} \leq \frac{1}{2}
\end{aligned}$$

if $\alpha = 1 + o(1)$ and n are sufficiently large. □

We note that the bound for the maximum number of balls in Claim 3.6 is essentially best possible, since the probability that the maximum number of balls is at least $(1-o(1)) \log n / \log \log n$ can be shown to be also at least $1/2$.

Example 3: Again, consider the situation that we have n balls and n bins, and each ball is placed in a bin chosen independently and uniformly at random.

Claim 3.7 *The probability that bin 1 has at least one ball is at least* $1/2$*.*

Proof. For any $i \in [n]$, let A_i be the event that ball i is placed in bin 1. Then, by Boole's inequality,

$$\begin{aligned}
\Pr[\text{bin 1 has at least one ball}] &= \Pr\left[\bigcup_{i \in [n]} A_i\right] \\
&\geq \sum_{1 \leq i \leq n} \Pr[A_i] - \sum_{1 \leq i < j \leq n} \Pr[A_i \cap A_j] \\
&= \sum_{1 \leq i \leq n} \frac{1}{n} - \sum_{1 \leq i < j \leq n} \frac{1}{n^2} \\
&= 1 - \binom{n}{2} \frac{1}{n^2} \geq 1 - \frac{1}{2} = \frac{1}{2} \ .
\end{aligned}$$

□

Observe that the probability bound in Claim 3.7 is not far away from the exact bound:

$$\Pr[\text{bin 1 has at least one ball}] = 1 - \left(1 - \frac{1}{n}\right)^n \overset{n\to\infty}{=} 1 - \frac{1}{\mathrm{e}} .$$

3.2.3 Conditional Probability

The *conditional probability* of event B assuming an event A with $\Pr[A] > 0$ is denoted by $\Pr[B \mid A]$ and defined as

$$\Pr[B \mid A] = \frac{\Pr[A \cap B]}{\Pr[A]} .$$

From this definition it follows that for all events A and B with $\Pr[A] > 0$,

$$\Pr[A \cap B] = \Pr[A] \cdot \Pr[B \mid A] . \tag{3.2}$$

This can be generalized as follows: If $A_1, \ldots, A_n$ are events with $\Pr[A_1 \cap \ldots \cap A_{n-1}] > 0$, then

$$\Pr[A_1 \cap \ldots \cap A_n] = \prod_{i=1}^{n} \Pr[A_i \mid A_1 \cap \ldots \cap A_{i-1}] .$$

Suppose that A and B are events with $\Pr[A] > 0$ and $\Pr[B] > 0$. Then, in addition to the equality (3.2), we have

$$\Pr[A \cap B] = \Pr[B] \cdot \Pr[A \mid B] . \tag{3.3}$$

From (3.2) and (3.3) we obtain *Bayes's formula*

$$\Pr[A \mid B] = \frac{\Pr[A] \cdot \Pr[B \mid A]}{\Pr[B]} .$$

Two events A and B are called *independent* if and only if

$$\Pr[B \mid A] = \Pr[B] .$$

Note that, due to Bayes's formula, in this case also $\Pr[A \mid B] = \Pr[A]$, that is, the independence property is *symmetric*.

If $\Pr[B \mid A] \neq \Pr[B]$, then A and B are said to be *correlated*. A and B are called

- *negatively correlated* if $\Pr[B \mid A] < \Pr[B]$ and
- *positively correlated* if $\Pr[B \mid A] > \Pr[B]$.

By Bayes's formula, all of these correlation properties are also symmetric.

As an example, any two disjoint events A and B with positive probabilities cannot be independent, since $\Pr[B \mid A] = 0$. However, they are always negatively correlated. Furthermore, they have the property that

$$\Pr[A \cup B] = \Pr[A] + \Pr[B] .$$

To illustrate negative and positive correlation, let us give a very simple example.

Example: Consider the problem of throwing n balls into n bins, where each ball is placed in a bin chosen independently and uniformly at random. Let A be the event that bin 1 has no ball, and let B be the event that some fixed bin $i > 1$ has no ball. For $n = 2$ it holds

$$\Pr[B \mid A] = 0 \leq \frac{1}{4} = \Pr[B] \ ,$$

and for $n = 3$ it holds

$$\Pr[B \mid A] = \frac{1}{8} \leq \frac{8}{27} = \Pr[B] \ .$$

Thus, A and B are negatively correlated for these n. (One can observe that this also holds for higher n.)

Now, let A be the event that bin 1 has exactly one ball, and let B be the event that some fixed bin $i > 1$ has exactly one ball. Since the expected number of balls in a bin is 1, one may assume that A does not influence B. However, for $n = 2$ it holds

$$\Pr[B \mid A] = 1 \geq \frac{1}{2} = \Pr[B] \ ,$$

and for $n = 3$ it holds

$$\Pr[B \mid A] = \frac{1}{2} \geq \frac{4}{9} = \Pr[B] \ .$$

Thus, A and B are positively correlated for these n. However, one can observe that $\Pr[B \mid A]$ tends to $\Pr[B]$ for $n \to \infty$.

If we define A and B to be events for which a bin has exactly $k > 1$ balls, one can show that A and B are, as in the first case, negatively correlated.

3.2.4 Random Variables

Any numerical function $X = X(\omega)$ defined on a sample space Ω may be called a *random variable*. In this thesis we will only consider real-valued random variables, i.e., functions of the form $X : \Omega \to \mathbb{R}$. It has been found convenient to separate the random variables into three categories: discrete, continuous, and mixed.

A *discrete* random variable is supposed to take only isolated values with nonzero probabilities. The number of values it is allowed to take may be infinite, but it is essential that they are *countable*. That is, it must be possible to arrange its values in a sequence so that the rth number is identifiable for any $r \in \mathbb{N}$. An *(absolutely) continuous* random variable X is one for which $\Pr[X \leq x]$ can be expressed as the integral

$$F_X(x) = \int_{-\infty}^{x} p_X(x)dx$$

of a function $p_X(x)$ commonly called the *probability density* of X. F_X is called the *distribution function* of X. A random variable is called *mixed* if it has both discrete and continuous parts.

In the following, we will only consider discrete random variables. A random variable X is called *non-negative* if $X(\omega) \geq 0$ for all $\omega \in \Omega$. For the special case that X maps elements in Ω to $\{0, 1\}$, X is called a *binary* or *Bernoulli* random variable. A binary random variable X is called an *indicator* of event A (denoted by I_A) if $X(\omega) = 1$ if and only if $\omega \in A$ for all $\omega \in \Omega$.

For any random variable X and any number $x \in \mathbb{R}$, we define $[X = x] = \{\omega \in \Omega : X(\omega) = x\}$. Instead of using set operations to express combinations of events associated with random variables, we will use logical expressions in the following, that is,

- instead of $\Pr[[X = x] \cap [Y = y]]$ we write $\Pr[X = x \wedge Y = y]$, and
- instead of $\Pr[[X = x] \cup [Y = y]]$ we write $\Pr[X = x \vee Y = y]$.

For any discrete random variable X and any $k \in \mathrm{IR}$, we define

$$\Pr[X \leq k] = \sum_{\ell \leq k} \Pr[X = \ell] \quad \text{and} \quad \Pr[X \geq k] = \sum_{\ell \geq k} \Pr[X = \ell] \ .$$

The function $p_X(k) = \Pr[X = k]$ is called the *probability distribution* of X, and the function $F_X(k) = \Pr[X \leq k]$ is called the *(cumulative) distribution function* of X. Furthermore, the function $G_X(k) = \Pr[X \geq k]$ is called the *survival distribution function* of X. For any two random variables X and Y, X is said to *(stochastically) dominate* Y (denoted by $X \succ Y$) if $G_X(k) \geq G_Y(k)$ for all $k \in \mathrm{IR}$.

3.2.5 Expectation

The most important measure used in combination with random variables is the expectation.

Definition 3.8 *Let $(\Omega, \mathcal{A}, p)$ denote an arbitrary probability space and $X : \Omega \to \mathrm{IR}$ be an arbitrary discrete function. Then the* expectation *of X is defined as*

$$\mathrm{E}[X] = \sum_{x \in \mathrm{IR}} x \cdot \Pr[X = x] \ . \tag{3.4}$$

Assume that the set $S = \{x_1, x_2, \ldots\}$ contains all values X can take, where $x_1 < x_2 < \ldots$. Let the function $\Delta_X : \mathrm{IR} \to \mathrm{IR}_+$ be defined as

$$\Delta_X(x) = \begin{cases} 0 & : \ x \notin S \\ x & : \ x = x_1 \\ x - x_{i-1} & : \ x = x_i \text{ for some } i \geq 2 \end{cases}$$

That is, $\Delta_X(x)$ provides the distance to the predecessor of x in S. If X is non-negative, it follows from equation (3.4) and the definition of Δ_X that

$$\mathrm{E}[X] = \sum_{x \geq 0} \Delta_X(x) \cdot \Pr[X \geq x] = \sum_{x \geq 0} \Delta_X(x) \cdot G_X(x) \ . \tag{3.5}$$

In the case that X is integer-valued, we can simplify this expression to

$$\mathrm{E}[X] = \sum_{x \in \mathrm{IN}} \Pr[X \geq x] = \sum_{x \in \mathrm{IN}} G_X(x) \ . \tag{3.6}$$

For binary random variables X it holds that

$$\mathrm{E}[X] = \Pr[X = 1] \ .$$

Basic properties

The following fact lists some basic properties of the expectation.

Fact 3.9 *For arbitrary random variables X and Y and any $c \in \mathrm{IR}$ it holds:*

- *If X is non-negative, then* $\mathrm{E}[X] \geq 0$.

- $|\mathrm{E}[X]| \leq \mathrm{E}[|X|]$.
- $\mathrm{E}[c \cdot X] = c \cdot \mathrm{E}[X]$.
- $\mathrm{E}[X+Y] = \mathrm{E}[X] + \mathrm{E}[Y]$.

Two random variables X and Y are called *independent* if, for all $x, y \in \mathbb{R}$,

$$\Pr[X = x \mid Y = y] = \Pr[X = x] \ .$$

Independent random variables have the following important property.

Claim 3.10 *If X and Y are independent random variables, then* $\mathrm{E}[X \cdot Y] = \mathrm{E}[X] \cdot \mathrm{E}[Y]$.

Proof. For all $x, y \in \mathbb{R}$, let A_x and B_y be the events with

$$A_x = \{\omega \in \Omega : \ X(\omega) = x\} \quad \text{and} \quad B_y = \{\omega \in \Omega : \ Y(\omega) = y\} \ .$$

Since X and Y are independent, $\Pr[A_x \cap B_y] = \Pr[A_x] \cdot \Pr[B_y]$ for all $x, y \in \mathbb{R}$. Recall that, for any event E, the indicator variable I_E is equal to 1 if and only if $\omega \in E$. As $X = \sum_x x \cdot I_{A_x}$ and $Y = \sum_y y \cdot I_{B_y}$, we obtain

$$\begin{aligned}
\mathrm{E}[X \cdot Y] &= \mathrm{E}\left[\left(\sum_{x \in \mathbb{R}} x \cdot I_{A_x}\right) \cdot \left(\sum_{y \in \mathbb{R}} y \cdot I_{B_y}\right)\right] \\
&= \mathrm{E}\left[\sum_{x,y \in \mathbb{R}} x \cdot y \cdot I_{A_x \cap B_y}\right] = \sum_{x,y \in \mathbb{R}} x \cdot y \cdot \Pr[A_x \cap B_y] \\
&= \sum_{x,y \in \mathbb{R}} x \cdot y \cdot \Pr[A_x] \cdot \Pr[B_y] \\
&= \left(\sum_{x \in \mathbb{R}} x \cdot \Pr[A_x]\right) \cdot \left(\sum_{y \in \mathbb{R}} y \cdot \Pr[B_y]\right) = \mathrm{E}[X] \cdot \mathrm{E}[Y] \ .
\end{aligned}$$

□

A set $X_1, \ldots, X_n$ of random variables is called *(mutually) independent* if, for all $i \in [n]$ and $S \subseteq [n] \setminus \{i\}$,

$$\Pr\left[X_i = x_i \mid \bigwedge_{j \in S} X_j = x_j\right] = \Pr[X_i = x_i] \tag{3.7}$$

for all $x_i \in \mathbb{R}$ and $x_j \in \mathbb{R}$ with $j \in S$. $X_1, \ldots, X_n$ are called *k-wise independent* if (3.7) holds for all $S \subseteq [n] \setminus \{i\}$ with $|S| \leq k$.

Modeling Dependence

The *conditional expectation* of a random variable Y with respect to an event A is defined by

$$\mathrm{E}[Y \mid A] = \sum_{y \in \mathbb{R}} y \cdot \Pr[Y = y \mid A] \ . \tag{3.8}$$

If the event A is $X = a$ for some random variable X, this equation defines a function f with

$$f(a) = \mathrm{E}[Y \mid X = a] \ .$$

Thus, $\mathrm{E}[Y \mid X]$ is a random variable, namely the random variable $f(X)$. (Observe that it does *not* hold in general that $\mathrm{E}[X \cdot Y] = \mathrm{E}[X] \cdot \mathrm{E}[Y \mid X]$. A way to study the relationship between $\mathrm{E}[X \cdot Y]$ and $\mathrm{E}[X]$ and $\mathrm{E}[Y]$ will be given in Section 3.2.6.)

A fundamental property of the conditional expectation is that

$$\mathrm{E}[Y] = \mathrm{E}[\mathrm{E}[Y \mid X]]$$

for any two random variables X and Y (which is easy to check). If Y is independent of X, then (3.8) implies that $\mathrm{E}[Y \mid X] = \mathrm{E}[Y]$. An important case for which $\mathrm{E}[Y]$ depends on X is given in the following definition.

Definition 3.11 *A sequence of random variables $X_0, X_1, \ldots$ is called a* martingale *if for all $i \geq 1$,*

$$\mathrm{E}[X_i \mid X_0, X_1, \ldots, X_{i-1}] = X_{i-1} \ .$$

Martingales will be considered in a later section.

Tail estimates

We start with a fact that contains some straightforward probability bounds.

Fact 3.12 *Let X be an arbitrary random variable. Then*

$$\Pr[X < \mathrm{E}[X]] < 1 \quad \textit{and} \quad \Pr[X > \mathrm{E}[X]] < 1 \ .$$

The next result provides a first, simple probability bound that depends on the deviation from the expected value. It has apparently first been used by Chebychev (which is why some authors call it Chebychev inequality [Shi96]), but it is commonly called *Markov inequality*.

Theorem 3.13 (Markov Inequality) *Let X be an arbitrary non-negative random variable. Then, for any $k > 0$,*

$$\Pr[X \geq k] \leq \frac{\mathrm{E}[X]}{k} \ .$$

Proof. Obviously,

$$\mathrm{E}[X] = \sum_{x \geq 0} x \cdot \Pr[X = x] \geq k \cdot \Pr[X \geq k] \ .$$

□

As we will show later, this inequality has a tremendous number of consequences. We will start with some simple examples.

Example 1: A directed graph G is called a *tournament* if for every pair of nodes u, v with $u \neq v$, either (u, v) or (v, u) is an edge in G. A *Hamiltonian path* in a directed graph G is a directed path that visits every node of G exactly once. We will show the following claim.

Claim 3.14 *There exists a tournament of size n with at least $n!/2^{n-1}$ Hamiltonian paths.*

Proof. Let K_n be the complete undirected graph of size n. We generate a random tournament T out of K_n by choosing a direction for each edge independently and uniformly at random. Let π be a permutation of the nodes in T. We define the binary random variable X_π to be 1 if and only if π defines a directed path in T. Clearly, for all $\pi \in S_n$, $\Pr[X_\pi = 1] = 2^{-(n-1)}$. Let X be the random variable counting the number of Hamiltonian paths in T, that is,

$$X = \sum_{\pi \in S_n} X_\pi \ .$$

Since $|S_n| = n!$, we obtain

$$\mathrm{E}[X] = \sum_{\pi \in S_n} \Pr[X_\pi = 1] = n!/2^{n-1} \ .$$

According to Fact 3.12, $\Pr[X < \mathrm{E}[X]] < 1$. Thus, there must exist a tournament with at least $n!/2^{n-1}$ Hamiltonian paths. □

Example 2 [Sch98]: Consider an arbitrary network G of size n, and let S_n denote the set of all permutations on $[n]$. For a permutation $\pi \in S_n$, let $R(G, \pi)$ be the minimum possible number of steps required to route packets in G according to π. Then the *routing number* R of G is defined as

$$R = \max_{\pi \in S_n} R(G, \pi) \ .$$

Let $\bar{R} = (1/|S_n|) \sum_{\pi \in S_n} R(G, \pi)$ denote the average number of steps to route a permutation in G using a best possible routing strategy. We intend to show the following claim.

Claim 3.15 *For any graph G, $\bar{R} = \Theta(R)$.*

Proof. It is obvious that $\bar{R} \leq R$ and therefore $\bar{R} = O(R)$. Hence, it remains to be shown that $\bar{R} = \Omega(R)$.

Consider any fixed permutation π. In order to bound the minimum number of steps to route π, we will use a probabilistic argument based on Valiant's trick (see [Val82]) by first sending the packets to random intermediate destinations before sending them to the destinations prescribed by π. Let X be a random variable denoting the minimum number of steps necessary to route a randomly chosen permutation. In this case, $\mathrm{E}[X] = \bar{R}$. According to the Markov inequality it holds:

$$\Pr[X \geq 3\bar{R}] \leq \tfrac{1}{3} \ .$$

Therefore, for a randomly chosen permutation φ for the intermediate destinations, it holds that the minimum number of steps to route the packets first to their intermediate destinations (prescribed by φ) and then to their final destinations (prescribed by π) is at least $6\bar{R}$ with probability at most $\frac{1}{3} + \frac{1}{3} < 1$. Hence, there exists an offline protocol that routes π in at most $6\bar{R}$ steps. Thus, $R \leq 6\bar{R}$, which completes the proof. □

3.2.6 Correlation

Two random variables X and Y are said to be *correlated* if and only if

$$\mathrm{E}[X \cdot Y] \neq \mathrm{E}[X] \cdot \mathrm{E}[Y] .$$

Otherwise, X and Y are called *uncorrelated*. Note that random variables may be uncorrelated, but not independent. Here is a simple example. Let the random variable X take the values 0, $\pi/2$ and π with probability $1/3$. Then $Y = \sin X$ and $Z = \cos X$ are uncorrelated. However, they are not independent, since

$$\Pr[Y = 1 \ \wedge \ Z = 1] = 0 \neq \frac{1}{9} = \Pr[Y = 1] \cdot \Pr[Z = 1] .$$

For the special case that X and Y are binary random variables, X and Y are uncorrelated if and only if X and Y are independent. Two random variables X and Y are called

- *negatively correlated* if $\mathrm{E}[X \cdot Y] \leq \mathrm{E}[X] \cdot \mathrm{E}[Y]$ and
- *positively correlated* if $\mathrm{E}[X \cdot Y] \geq \mathrm{E}[X] \cdot \mathrm{E}[Y]$.

In general, a collection of random variables $X_1, \ldots, X_n$ is called *(mutually) uncorrelated* if

$$\mathrm{E}\left[\prod_{i \in S} X_i\right] = \prod_{i \in S} \mathrm{E}[X_i] \tag{3.9}$$

for every $S \subseteq [n]$. $X_1, \ldots, X_n$ are called *(mutually) negatively correlated* or *(mutually) positively correlated* if instead of "$=$" we have "$\leq$" or "$\geq$", respectively. Furthermore, $X_1, \ldots, X_n$ are called *k-wise uncorrelated* if (3.9) holds for every $S \subseteq [n]$ with $|S| \leq k$. k-wise negative and positive correlation is defined accordingly.

Panconesi and Srinivasan defined in [PS92] the notion of self-weakening random variables, which can be seen as a generalization to the notion of negatively correlated random variables. A collection of random variables $X_1, \ldots, X_n$ is called *self-weakening with parameter λ* if

$$\mathrm{E}\left[\prod_{i \in S} X_i\right] \leq \lambda \prod_{i \in S} \mathrm{E}[X_i] \tag{3.10}$$

for every $S \subseteq [n]$.

In the following, we present two methods of proving correlation properties. The first uses stochastic domination (recall the definition in Section 3.2.4).

Claim 3.16 *Consider any two non-negative random variables X and Y.*

(1) If $Y \succ (Y \mid X \geq x)$ for all $x \in \mathbb{R}$, then X and Y are negatively correlated.

(2) If $Y \prec (Y \mid X \geq x)$ for all $x \in \mathbb{R}$, then X and Y are positively correlated.

Proof. We only prove (1). The proof for (2) is analogous. For all $x, y \in \mathbb{R}$, let A_x and B_y be the events with

$$A_x = \{\omega \in \Omega : \ X(\omega) \geq x\} \quad \text{and} \quad B_y = \{\omega \in \Omega : \ Y(\omega) \geq y\} .$$

Since $Y \succ (Y \mid X \geq x)$ for all $x \in \mathbb{R}$, $\Pr[Y \geq y] \geq \Pr[Y \geq y \mid X \geq x]$ for all $x, y \in \mathbb{R}$. Hence,

$$\Pr[A_x \cap B_y] = \Pr[A_x] \cdot \Pr[B_y \mid A_x] \leq \Pr[A_x] \cdot \Pr[B_y]$$

for all $x, y \in \mathbb{R}$. It is easy to verify that X is equal to $\sum_{x \geq 0} \Delta_X(x) I_{A_x}$ (see equation (3.5)). Thus,

$$\begin{aligned}
\mathrm{E}[X \cdot Y] &= \mathrm{E}\left[\left(\sum_{x \geq 0} \Delta_X(x) \cdot I_{A_x}\right) \cdot \left(\sum_{y \geq 0} \Delta_Y(y) \cdot I_{B_y}\right)\right] \\
&= \mathrm{E}\left[\sum_{x,y \geq 0} \Delta_X(x) \cdot \Delta_Y(y) \cdot I_{A_x \cap B_y}\right] = \sum_{x,y \geq 0} \Delta_X(x) \cdot \Delta_Y(y) \cdot \Pr[A_x \cap B_y] \\
&\leq \sum_{x,y \geq 0} \Delta_X(x) \cdot \Delta_Y(y) \cdot \Pr[A_x] \cdot \Pr[B_y] \\
&= \left(\sum_{x \geq 0} \Delta_X(x) \cdot \Pr[A_x]\right) \cdot \left(\sum_{y \geq 0} \Delta_Y(y) \cdot \Pr[B_y]\right) = \mathrm{E}[X] \cdot \mathrm{E}[Y] \; ,
\end{aligned}$$

which implies that X and Y are negatively correlated. □

Another important tool for the study of the relationship between $\mathrm{E}[X \cdot Y]$ and $\mathrm{E}[X] \cdot \mathrm{E}[Y]$ is the so-called FKG inequality. Before we present it, we introduce some notation.

A *lattice* is a partially ordered set in which every two elements x and y have a unique minimal upper bound, denoted by $x \vee y$ and called the *join* of x and y, and a unique maximal lower bound, denoted by $x \wedge y$ and called the *meet* of x and y. A lattice is *distributive* if for all $x, y, z \in L$,

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$

or, equivalently, if for all $x, y, z \in L$,

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z) \; .$$

To give an example, any subset L of Ω which is closed under the union and intersection operations is a distributive lattice. Here, the join of two members $A, B \in L$ is simply their union $A \cup B$ and their meet is the intersection $A \cap B$.

A function $\mu : L \to \mathbb{R}_+$, where L is a distributive lattice, is called *log-supermodular* if

$$\mu(x)\mu(y) \leq \mu(x \vee y)\mu(x \wedge y)$$

for all $x, y \in L$. A function $f : L \to \mathbb{R}$ is *non-decreasing* if $f(x) \leq f(y)$ whenever $x \leq y$ and is *non-increasing* if $f(x) \geq f(y)$ whenever $x \leq y$.

Motivated by a problem from statistical mechanics, Fortuin, Kasteleyn, and Ginibre [FGK71] proved the following useful inequality.

Theorem 3.17 (FKG Inequality) *Let L be a finite distributive lattice and let $\mu : L \to \mathbb{R}_+$ be a log-supermodular function. Then, if $f, g : L \to \mathbb{R}_+$ are both non-decreasing or non-increasing, we have*

$$\left(\sum_{x \in L} \mu(x) f(x)\right) \cdot \left(\sum_{x \in L} \mu(x) g(x)\right) \leq \left(\sum_{x \in L} \mu(x) f(x) g(x)\right) \cdot \left(\sum_{x \in L} \mu(x)\right) \; .$$

If one of the functions is non-decreasing and the other is non-increasing, then the reverse inequality holds.

It is helpful to view μ as a measure on L. Assuming μ is not identically 0, we can define for any $f : L \to \mathbb{R}$ its expectation by

$$\mathrm{E}_\mu[f] = \frac{\sum_{x \in L} \mu(x) f(x)}{\sum_{x \in L} \mu(x)} .$$

In this notation, the FKG inequality asserts that for any log-supermodular μ and any two functions $f, g : L \to \mathbb{R}_+$,

$$\mathrm{E}_\mu[f] \cdot \mathrm{E}_\mu[g] \leq \mathrm{E}_\mu[f \cdot g]$$

if f, g are both non-decreasing or non-increasing, and

$$\mathrm{E}_\mu[f] \cdot \mathrm{E}_\mu[g] \geq \mathrm{E}_\mu[f \cdot g]$$

if one of the functions is non-decreasing and the other is non-increasing.

To demonstrate the usefulness of Claim 3.16 and the FKG inequality, we present a simple example.

Example [DR94]: Suppose we throw m balls into n bins uniformly and independently at random, for positive integers m and n. Let X_i be the random variable denoting the number of balls in the ith bin.

A possible *configuration* of the experiment can be represented by a vector $\boldsymbol{a} = (a_1, \ldots, a_m)$, with $a_i \in [n]$ for every $i \in [m]$. This is the configuration where ball i goes into bin a_i for every $i \in [m]$. Define the lattice L to be the set of all such configurations ordered component-wise:

$$\boldsymbol{a} \leq_L \boldsymbol{b} \quad \Leftrightarrow \quad a_i \leq b_i \text{ for every } i \in [m].$$

It turns out that this in fact defines a distributive lattice, with join and meet given by the following equation on the components:

$$(\boldsymbol{a} \wedge \boldsymbol{b})_i = \max[a_i, b_i] \quad \text{and} \quad (\boldsymbol{a} \vee \boldsymbol{b})_i = \min[a_i, b_i] .$$

Distributivity follows because of the following property of the integers:

$$\begin{aligned} \min[a, \max[b, c]] &= \max[\min[a, b], \min[a, c]] , \\ \max[a, \min[b, c]] &= \min[\max[a, b], \max[a, c]] . \end{aligned}$$

Now define $\mu : L \to \mathbb{R}_+$ by $\mu(\boldsymbol{a}) = 1/n^m$ for every $\boldsymbol{a} \in L$. So defined, μ is trivially log-supermodular, and crucially, it makes each configuration equally likely, representing the fact that the balls are thrown uniformly and independently at random into the bins.

For any configuration $\boldsymbol{a}$ and any $i \in [n]$, we introduce the random variable

$$X_i(\boldsymbol{a}) = |\{j : \; a_j = i\}| .$$

This gives the number of balls in the ith bin in configuration $\boldsymbol{a}$. Let $I, J \subseteq [n]$ be two index sets such that either $I \cap J = \emptyset$ or $I \cup J = [n]$. With no loss of generality, we can arrange it by renumbering, so that $J = \{1, \ldots, |J|\}$ and $I = \{n - |I| + 1, \ldots, n\}$. Let t_I and t_J be arbitrary non-negative integers. Define $f, g : L \to \mathbb{R}_+$ as indicator functions by

$$f(\boldsymbol{a}) = \begin{cases} 1 & : \; \sum_{i \in I} X_i(\boldsymbol{a}) \geq t_I \\ 0 & : \; \text{otherwise} \end{cases}$$

and

$$g(\boldsymbol{a}) = \begin{cases} 1 & : \quad \sum_{j\in J} X_j(\boldsymbol{a}) \geq t_J \\ 0 & : \quad \text{otherwise} \end{cases}$$

The definition of the lattice order ensures that f is non-decreasing while g is non-increasing. Also, note that

$$\begin{aligned} \sum_{\boldsymbol{a}\in L} \mu(\boldsymbol{a}) f(\boldsymbol{a}) &= \frac{1}{n^m} \cdot |\{\boldsymbol{a} \in L : \sum_{i\in I} X_i(\boldsymbol{a}) \geq t_I\}| \\ &= \Pr[\sum_{i\in I} X_i \geq t_I] \ . \end{aligned}$$

Similarly,

$$\begin{aligned} \sum_{\boldsymbol{a}\in L} \mu(\boldsymbol{a}) g(\boldsymbol{a}) &= \Pr[\sum_{j\in J} X_j \geq t_J] \quad \text{and} \\ \sum_{\boldsymbol{a}\in L} \mu(\boldsymbol{a}) f(\boldsymbol{a}) g(\boldsymbol{a}) &= \Pr[\sum_{i\in I} X_i \geq t_I \ \wedge \ \sum_{j\in J} X_j \geq t_J] \ . \end{aligned}$$

Applying the FKG inequality, we obtain the following correlation inequality on the random variables X_i:

Claim 3.18 *Let $I, J \subseteq [n]$ be index sets such that $I \cap J = \emptyset$ or $I \cup J = [n]$, and let t_I, t_J be arbitrary non-negative integers. Then*

$$\Pr[\sum_{i\in I} X_i \geq t_I \mid \sum_{j\in J} X_j \geq t_J] \leq \Pr[\sum_{i\in I} X_i \geq t_I] \ .$$

This claim and Claim 3.16 together imply that the X_i's are pairwise negatively correlated.

3.2.7 Variance

The *variance* (or *dispersion*) of a random variable X, denoted by $\mathrm{V}[X]$, is defined as

$$\mathrm{V}[X] = \mathrm{E}[(X - \mathrm{E}[X])^2] \ .$$

The number $\sigma = \sqrt{\mathrm{V}[X]}$ is called the *standard deviation* of X.

Basic properties

Since

$$\mathrm{E}[(X - \mathrm{E}[X])^2] = \mathrm{E}[X^2 - 2X \cdot \mathrm{E}[X] + (\mathrm{E}[X])^2] = \mathrm{E}[X^2] - (\mathrm{E}[X])^2 \ ,$$

we have

$$\mathrm{V}[X] = \mathrm{E}[X^2] - (\mathrm{E}[X])^2 \ .$$

As $\mathrm{E}[X^2] \geq \mathrm{E}[X]^2$ for any random variable X, $\mathrm{V}[X] \geq 0$. Furthermore, it follows from the definition that

$$\mathrm{V}[a + bX] = b^2 \mathrm{V}[X]$$

for any two constants a and b.

Let X and Y be random variables. Then

$$\begin{aligned} \mathrm{V}[X + Y] &= \mathrm{E}[((X - \mathrm{E}[X]) + (Y - \mathrm{E}[Y]))^2] \\ &= \mathrm{V}[X] + \mathrm{V}[Y] + 2\mathrm{E}[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])] \ . \end{aligned}$$

$\mathrm{E}[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])]$ is called the *covariance* of X and Y and denoted by $\mathrm{cov}(X, Y)$. We observe immediately that if X and Y are independent, so are $X - \mathrm{E}[X]$ and $Y - \mathrm{E}[Y]$. Consequently, in this case,

$$\mathrm{cov}(X, Y) = \mathrm{E}[X - \mathrm{E}[X]] \cdot \mathrm{E}[Y - \mathrm{E}[Y]] = 0 \ .$$

Thus, if X and Y are independent, then

$$\mathrm{V}[X + Y] = \mathrm{V}[X] + \mathrm{V}[Y] \ .$$

In general, if $X_1, \ldots, X_n$ are independent, then

$$\mathrm{V}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathrm{V}[X_i] + 2\sum_{i<j} \mathrm{cov}(X_i, X_j) = \sum_{i=1}^{n} \mathrm{V}[X_i] \ .$$

Since $\mathrm{cov}(X_i, X_j) = 0$ if and only if X_i and X_j are uncorrelated, the equality already holds for pairwise uncorrelated $X_1, \ldots, X_n$. Furthermore, the following fact holds.

Fact 3.19 *For any two random variables X and Y,*

- $\mathrm{cov}(X, Y) = 0$ *if and only if X and Y are uncorrelated,*
- $\mathrm{cov}(X, Y) \leq 0$ *if and only if X and Y are negatively correlated, and*
- $\mathrm{cov}(X, Y) \geq 0$ *if and only if X and Y are positively correlated.*

Because of this fact, the correlation property is often defined by the use of cov.

Modeling Dependence

The *conditional variance* of a random variable Y with respect to an event A is defined by

$$\mathrm{V}[Y \mid A] = \mathrm{E}[(Y - \mathrm{E}[Y \mid A])^2 \mid A] \ . \tag{3.11}$$

If the event A is $X = a$ for some random variable X, this equation defines a function f with

$$f(a) = \mathrm{V}[Y \mid X = a] \ .$$

Thus, $\mathrm{V}[Y \mid X]$ is a random variable, namely the random variable $f(X)$. If Y is independent of X then (3.11) implies that $\mathrm{V}[Y \mid X] = \mathrm{V}[Y]$.

A fundamental property of the conditional variance is that

$$\mathrm{V}[Y] = \mathrm{E}[\mathrm{V}[Y \mid A]] + \mathrm{V}[\mathrm{E}[Y \mid A]]$$

for any random variable Y and any event A (which is easy to check). Thus, observe that (in contrast to $\mathrm{E}[\mathrm{E}[Y \mid X]] = \mathrm{E}[Y]$) it does *not* hold in general that $\mathrm{E}[\mathrm{V}[Y \mid X]] = \mathrm{V}[Y]$.

Conditional variances have rarely been considered, and so far no important tail estimates exist that use them. Hence, we will not consider them any further in this thesis.

Tail estimates

The most well-known tail estimate that uses the variance of a random variable is due to Chebychev.

Theorem 3.20 (Chebychev Inequality) *Let X be an arbitrary random variable. Then, for every $k > 0$,*

$$\Pr[|X - \mathrm{E}[X]| \geq k] \leq \frac{\mathrm{V}[X]}{k^2} .$$

Proof. From the Markov inequality it follows for all random variables X that

$$\Pr[|X| \geq k] = \Pr[X^2 \geq k^2] \leq \mathrm{E}[X^2]/k^2 .$$

Replacing $|X|$ by $|X - \mathrm{E}[X]|$ yields the theorem. □

Let us apply this theorem to some simple examples.

Example 1: Consider again the example in Section 3.2.6, that is, we throw m balls into n bins uniformly and independently at random. Let the binary random variable X_i be 1 if and only if at least one ball falls into bin i, and let $X = \sum_{i=1}^{n} X_i$ count the number of non-empty bins. Since we know from the example above that the X_i's are pairwise negatively correlated, we have

$$\mathrm{V}[X] \leq \sum_{i=1}^{n} \mathrm{V}[X_i] .$$

Thus, we only have to calculate the variance of each X_i. It is not difficult to see that

$$\begin{aligned} \mathrm{E}[X_i^2] &= \mathrm{E}[X_i] = \sum_{j=1}^{m} \binom{m}{j} \left(\frac{1}{n}\right)^j \left(1 - \frac{1}{n}\right)^{m-j} \\ &= 1 - F(n,m) . \end{aligned}$$

where $F(n,m) = (1 - 1/n)^m$. Hence,

$$\begin{aligned} \mathrm{V}[X_i] &= \mathrm{E}[X_i^2] - \mathrm{E}[X_i]^2 = (1 - F(n,m)) - (1 - F(n,m))^2 \\ &= F(n,m)(1 - F(n,m)) . \end{aligned}$$

Therefore, $\mathrm{V}[X] \leq n \cdot F(n,m)(1 - F(n,m))$. Moreover,

$$\mathrm{E}[X] = \sum_{i=1}^{n} \mathrm{E}[X_i] = n \cdot (1 - F(n,m)) .$$

Suppose that $m = n$ and n is infinitely large. Then $F(n,m) = 1/\mathrm{e}$ according to Section 3.1.2. In this case, $\mathrm{E}[X] = n(1 - 1/\mathrm{e})$ and $\mathrm{V}[X] \leq n/\mathrm{e}$. From the Chebychev Inequality it follows for all $k > 0$ that

$$\Pr\left[|X - n(1 - 1/\mathrm{e})| \geq k\sqrt{n/\mathrm{e}}\right] \leq \frac{1}{k^2}$$

Example 2: (The Coupon Collector Problem)
Suppose that we have n different bins. In each iteration we throw one ball into a bin chosen uniformly at random. The problem is to bound the number of time steps until every bin has at least one ball.

Let T be the random variable that counts the number of time steps until every bin has at least one ball. Our aim is to compute the expected value and the variance of T, so that we can use Chebychev's inequality.

Let the random variable Y_i count the number of time steps that have to pass by until a ball lands in an empty bin, assuming i bins are already filled with at least one ball. Then $T = \sum_{i=0}^{n-1} Y_i$. Clearly, for every $i \in \{0, \ldots, n-1\}$ we have

$$\Pr[Y_i = k] = p_i^{k-1}(1-p_i)$$

for every $k \in \mathbb{N}$, where $p_i = i/n$. Thus,

$$\begin{aligned}
\mathrm{E}[Y_i] &= \sum_{k=1}^{\infty} k \cdot p_i^{k-1}(1-p_i) = (1-p_i)\left(\sum_{k=0}^{\infty} p_i^k\right)^2 \\
&= (1-p_i) \cdot \frac{1}{(1-p_i)^2} = \frac{n}{n-i} \; .
\end{aligned}$$

and

$$\mathrm{E}[T] = \sum_{i=0}^{n-1} \mathrm{E}[Y_i] = \sum_{i=0}^{n-1} \frac{n}{n-i} = n \sum_{j=1}^{n} \frac{1}{j} = n \cdot H_n \; .$$

Next we calculate the variance of each Y_i. It holds that

$$\begin{aligned}
\mathrm{V}[Y_i] &= \mathrm{E}[Y_i^2] - \mathrm{E}[Y_i]^2 = \mathrm{E}[Y_i(Y_i-1)] + \mathrm{E}[Y_i] - \mathrm{E}[Y_i]^2 \\
&= \frac{2p_i}{(1-p_i)^2} + \frac{1}{1-p_i} - \frac{1}{(1-p_i)^2} \\
&= \frac{p_i}{(1-p_i)^2} \; .
\end{aligned}$$

Note that the random variables Y_i are independent. Therefore,

$$\begin{aligned}
\mathrm{V}[T] &= \mathrm{V}\left[\sum_{i=0}^{n-1} Y_i\right] = \sum_{i=0}^{n-1} \mathrm{V}[Y_i] \\
&= \sum_{i=0}^{n-1} \frac{i}{n} \cdot \frac{n^2}{(n-i)^2} = n \sum_{j=1}^{n} \frac{n-j}{j^2} \\
&= n^2 \sum_{j=1}^{n} \frac{1}{j^2} - n \sum_{j=1}^{n} \frac{1}{j} = n^2 \cdot \frac{\pi^2}{6} - n \cdot H_n \\
&\leq n^2 \cdot \frac{\pi^2}{6} \; .
\end{aligned}$$

It follows from the Chebychev inequality that for all $k > 0$,

$$\Pr[|T - n \cdot H_n| \geq k \cdot \pi n/\sqrt{6}] \leq \frac{1}{k^2} \; .$$

This bound is quite weak. In the following we will introduce techniques that will allow to obtain much better tail estimates.

3.2.8 Higher Order Moments

Expectations of powers of random variables are called *moments* and play an essential role in the investigations of probability theory. For us they will be of particular interest in the study of sums of random variables. Given a random variable X, the following expected values of functions of X are often studied:

- $\mathrm{E}[X^k]$, $k \in \mathbb{N}$: kth *moment* of X
- $\mathrm{E}[|X|^k]$, $k \in \mathbb{N}$: kth *absolute moment* of X
- $\mathrm{E}[|X - \mathrm{E}[X]|^k]$, $k \in \mathbb{N}$: kth *absolute central moment* of X
- $\mathrm{E}[X^{(k)}]$, $k \in \mathbb{N}$ where $X^{(k)} = X(X-1)\cdots(X-k+1)$: kth *(descending) factorial moment* of X
- $\mathrm{E}[X^{[k]}]$, $k \in \mathbb{N}$ where $X^{[k]} = X(X+1)\cdots(X+k-1)$: kth *ascending factorial moment* of X
- $\mathrm{E}[\mathrm{e}^{h\cdot X}]$, $h > 0$: *exponential moment* of X

According to Section 3.1.2, these moments obey the following identities:

- For every $h \in \mathbb{R}$,
$$\mathrm{E}[\mathrm{e}^{h\cdot X}] = \sum_{i=0}^{\infty} \frac{h^i \cdot \mathrm{E}[X^i]}{i!} .$$
- For every $r \in \mathbb{N}$,
$$\mathrm{E}[X^{(r)}] = \sum_{i=0}^{r} s_{r,i} \cdot \mathrm{E}[X^i] \quad \text{and} \quad \mathrm{E}[X^r] = \sum_{i=0}^{r} S_{r,i} \cdot \mathrm{E}[X^{(i)}] .$$
- For every $r \in \mathbb{N}$,
$$\mathrm{E}[X^{[r]}] = \sum_{i=0}^{r} |s_{r,i}| \cdot \mathrm{E}[X^i] \quad \text{and} \quad \mathrm{E}[X^r] = \sum_{i=0}^{r} (-1)^{r-i} S_{r,i} \cdot \mathrm{E}[X^{[i]}] .$$

These identities allow us to express moments in terms of other moments.

Basic properties

In the following we list a few properties of moments stated in [Loé77]. Let k be any positive integer. While a kth moment may not exist, absolute moments always exist but may be infinite. On the other hand, if the kth absolute moment of X is finite, then its kth moment exists and is finite; and conversely. Furthermore, since $|X|^r \leq 1 + |X|^s$ for all $0 < r < s$, finiteness of a moment of X implies existence and finiteness of all moments of X of lower order.

When applying the elementary inequality
$$|a + b|^r \leq 2^{r-1}(|a|^r + |b|^r)$$
for all $r \in \mathbb{N}$, replacing a by X and b by Y and taking expectations on both sides, we obtain the following inequality:
$$\mathrm{E}[|X + Y|^r] \leq 2^{r-1}(\mathrm{E}[|X|^r] + \mathrm{E}[|Y|^r]) \quad \text{for all } r \in \mathbb{N}.$$

This inequality shows that if the rth absolute moments of X and Y exist and are finite, so is the rth absolute moment of $X + Y$. The *Hölder inequality* gives a relation for products of random variables:

$$\mathrm{E}[|X \cdot Y|] \leq \mathrm{E}[|X|^r]^{1/r} \cdot \mathrm{E}[|Y|^s]^{1/s} \quad \text{for all } r, s \text{ with } r > 1 \text{ and } \tfrac{1}{r} + \tfrac{1}{s} = 1.$$

Hölder's inequality with $r = s = 2$ is called the *Schwartz inequality*:

$$\mathrm{E}[|X \cdot Y|]^2 \leq \mathrm{E}[X^2] \cdot \mathrm{E}[Y^2] \ .$$

Furthermore, the *Minkowski inequality* states that

$$\mathrm{E}[|X + Y|^r]^{1/r} \leq \mathrm{E}[|X|^r]^{1/r} + \mathrm{E}[|Y|^r]^{1/r} \quad \text{for all } r \in \mathrm{I\!N}.$$

A function f is said to be *convex* on $I \in \mathrm{I\!R}$ if, for every pair of points $x, x' \in I$,

$$f\left(\frac{x + x'}{2}\right) \leq \frac{1}{2} f(x) + \frac{1}{2} f(x') \ .$$

If f is twice differentiable on I, then the convexity property is equivalent to $f'' \geq 0$ on I. *Jensen's inequality* states that if f is convex and $\mathrm{E}[X]$ is finite, then

$$f(\mathrm{E}[X]) \leq \mathrm{E}[f(X)] \ .$$

Since $f(x) = x^r$ is convex for all $r \in \mathrm{I\!N}$, this implies that $\mathrm{E}[|X|] \leq \mathrm{E}[|X|^r]^{1/r}$, which is a special case of the Hölder inequality.

The following fact contains one more Chebychev inequality (see, e.g., [Pet95]).

Fact 3.21 (Chebychev) *Let (a, b) be any (finite or infinite) interval on $\mathrm{I\!R}$. Let u and v be both non-decreasing or both non-increasing functions defined on (a, b). Let X be a random variable taking values only from the interval (a, b). Then*

$$\mathrm{E}[u(X)] \cdot \mathrm{E}[v(X)] \leq \mathrm{E}[u(X) \cdot v(X)]$$

if these means exist. If one of u and v is non-decreasing and the other is non-increasing, then the reverse inequality holds.

Modeling Dependence

Next we use higher moments to define the notion of *association*, which is in some situations more useful than the correlation property. We present here a definition that is not as strict as the definitions in [JP83, DR94], but it suffices for all situations studied in this thesis.

Two variables X and Y are called *associated* if there are $k, \ell \in \mathrm{I\!N}$ with

$$\mathrm{E}[X^k \cdot Y^\ell] \neq \mathrm{E}[X^k] \cdot \mathrm{E}[Y^\ell]$$

or, equivalently, $\mathrm{cov}(X^k, Y^\ell) \neq 0$. Two variables X and Y are called *negatively associated* if, for all $k, \ell \in \mathrm{I\!N}$, $\mathrm{cov}(X^k, Y^\ell) \leq 0$ and are called *positively associated* if, for all $k, \ell \in \mathrm{I\!N}$, $\mathrm{cov}(X^k, Y^\ell) \geq 0$. Clearly, all association properties are symmetric.

In general, a collection of random variables $X_1, \ldots, X_n$ is called *(mutually) non-associated* if

$$\mathrm{E}\left[\prod_{j \in S} X_j^{k_j}\right] = \prod_{j \in S} \mathrm{E}[X_j^{k_j}]$$

for any $S \subseteq [n]$ and any $k_j \in \mathbb{N}$ with $j \in S$. Furthermore, $X_1, \ldots, X_n$ are called *k-wise non-associated* if the property above holds for every S with $|S| \leq k$.

In the following, we list some basic properties of associated random variables. For two binary random variables X and Y it is necessary and sufficient that $\mathrm{cov}(X,Y) \leq 0$ (resp. $\mathrm{cov}(X,Y) \geq 0$) for X and Y to be negatively (resp. positively) associated.

Claim 3.22 *Consider any two non-negative random variables X and Y.*

(1) If $Y \succ (Y|X \geq x)$ for all $x \in \mathbb{R}_+$, then X and Y are negatively associated.

(2) If $(Y|X \geq x) \succ Y$ for all $x \in \mathbb{R}_+$, then X and Y are positively associated.

Proof. We only prove (1). The proof for (2) is analogous. Consider any fixed $k, \ell \in \mathbb{N}$. For all $x, y \in \mathbb{R}_+$, let A_x and B_y be events with

$$A_x = \{\omega \in \Omega : \ X(\omega)^k \geq x\} \quad \text{and} \quad B_y = \{\omega \in \Omega : \ Y(\omega)^\ell \geq y\} \ .$$

Since $Y \succ (Y|X \geq x)$ for all $x \in \mathbb{R}_+$, $\Pr[Y \geq y] \geq \Pr[Y \geq y \mid X \geq x]$ for all $x, y \in \mathbb{R}_+$, and therefore, $\Pr[Y^\ell \geq y] \geq \Pr[Y^\ell \geq y \mid X^k \geq x]$ for all $x, y \in \mathbb{R}_+$. Hence,

$$\Pr[A_x \cap B_y] = \Pr[A_x] \cdot \Pr[B_y \mid A_x] \leq \Pr[A_x] \cdot \Pr[B_y]$$

for all $x, y \in \mathbb{R}$. The rest of the proof is identical to the proof of Claim 3.16. □

As we will see, our notion of association is strong enough to replace in the following all requirements that a collection of random variables be mutually independent by the requirement that this collection be mutually non-associated. However, note that the fact that a collection of random variables is non-associated does not imply that they are independent. If, for instance, two random variables X and Y are independent, then $f(X)$ and $g(Y)$ are independent for any two functions f and g. Thus, $\mathrm{cov}(f(X), g(Y)) = 0$ for all functions f and g, which may not be the case if X and Y are just known to be non-associated. However, we will show that for certain random variables and certain functions this is true. For this we need the following well-known theorem.

Theorem 3.23 (Weierstrass) *For every continuous function $f : [0,1] \to \mathbb{R}$ and every $\epsilon > 0$ there is a polynomial p such that $|f(x) - p(x)| < \epsilon$ for every $x \in [0,1]$.*

Theorem 3.23 allows to prove the following result.

Theorem 3.24 *For any two discrete, non-associated random variables X and Y and any two (in the ranges of X and Y) non-negative, continuous functions f and g with finite $\mathrm{E}[f(X)]$, $\mathrm{E}[g(Y)]$, and $\mathrm{E}[f(X)g(Y)]$,*

$$\mathrm{E}[f(X)g(Y)] = \mathrm{E}[f(X)] \cdot \mathrm{E}[g(Y)] \ .$$

Proof. For any discrete random variable X and any (finite or infinite) set $S \subseteq \mathbb{R}$, we define $\mathrm{E}_S[X]$ as

$$\mathrm{E}_S[X] = \sum_{x \in S} x \cdot \Pr[X = x] \ .$$

Now let X, Y, f, and g be chosen as in Theorem 3.24. Consider any fixed $\epsilon > 0$. Since f and g are non-negative and $\mathrm{E}[f(X)]$, $\mathrm{E}[g(Y)]$, and $\mathrm{E}[f(X)g(Y)]$ are finite, there is is a finite interval $I \subseteq \mathbb{R}$ such that

$$\mathrm{E}_{\mathbb{R}\setminus I}[f(X)] < \epsilon\ , \quad \mathrm{E}_{\mathbb{R}\setminus I}[g(Y)] < \epsilon\ , \quad \text{and} \quad \mathrm{E}_{\mathbb{R}\setminus I}[f(X)g(Y)] < \epsilon\ . \tag{3.12}$$

As f and g are continuous in I, there must be two polynomials p and q according to Theorem 3.23 with $|f(z) - p(z)| < \epsilon$ and $|g(z) - q(z)| < \epsilon$ for every $z \in I$. Since f and g are non-negative,

$$f(x)g(y) \leq (p(x) + \epsilon)(q(y) + \epsilon)$$

for all $x, y \in I$. Hence,

$$\begin{aligned} \mathrm{E}_I[f(X)g(Y)] &\leq \mathrm{E}_I[(p(X) + \epsilon)(q(Y) + \epsilon)] \\ &= (\mathrm{E}_I[p(X)] + \epsilon)(\mathrm{E}_I[q(Y)] + \epsilon)\ , \end{aligned} \tag{3.13}$$

where the equality follows from the fact that X and Y are non-associated. Furthermore, since $f(z) \geq p(z) - \epsilon$ and $g(z) \geq q(z) - \epsilon$ for every $z \in I$,

$$\mathrm{E}_I[f(X)] \geq \mathrm{E}_I[p(X)] - \epsilon \quad \text{and} \quad \mathrm{E}_I[g(Y)] \geq \mathrm{E}_I[q(Y)] - \epsilon\ . \tag{3.14}$$

Combining (3.13) and (3.14) yields

$$\mathrm{E}_I[f(X)g(Y)] \leq (\mathrm{E}_I[f(X)] + 2\epsilon)(\mathrm{E}_I[g(X)] + 2\epsilon)\ . \tag{3.15}$$

From (3.12) it follows that

$$\begin{aligned} \mathrm{E}[f(X)g(Y)] &\leq \mathrm{E}_I[f(X)g(Y)] + \epsilon \\ &\leq (\mathrm{E}_I[f(X)] + 2\epsilon)(\mathrm{E}_I[g(X)] + 2\epsilon) + \epsilon \\ &\leq (\mathrm{E}[f(X)] + 2\epsilon)(\mathrm{E}[g(X)] + 2\epsilon) + \epsilon\ . \end{aligned} \tag{3.16}$$

On the other hand,

$$f(x)g(y) \geq (p(x) - \epsilon)(q(y) - \epsilon) - 4\epsilon^2$$

for all $x, y \in I$. (The $4\epsilon^2$ is due to the fact that, in the worst case, $f(x) = g(y) = 0$ and $p(x) = q(y) = -\epsilon$, resulting in $f(x)g(y) = (p(x) - \epsilon)(q(y) - \epsilon) - 4\epsilon^2$.) Thus, since X and Y are non-associated,

$$\begin{aligned} \mathrm{E}_I[f(X)g(Y)] &\geq (\mathrm{E}_I[p(X)] - \epsilon)(\mathrm{E}_I[q(Y)] - \epsilon) - 4\epsilon^2 \\ &\geq (\mathrm{E}_I[f(X)] - 2\epsilon)(\mathrm{E}_I[g(Y)] - 2\epsilon) - 4\epsilon^2\ . \end{aligned} \tag{3.17}$$

From (3.12) it follows that

$$\begin{aligned} \mathrm{E}[f(X)g(Y)] &\geq \mathrm{E}_I[f(X)g(Y)] \\ &\geq (\mathrm{E}_I[f(X)] - 2\epsilon)(\mathrm{E}_I[g(Y)] - 2\epsilon) - 4\epsilon^2 \\ &\geq (\mathrm{E}[f(X)] - 3\epsilon)(\mathrm{E}[g(Y)] - 3\epsilon) - 4\epsilon^2\ . \end{aligned} \tag{3.18}$$

Combining (3.16) and (3.18) yields

$$\mathrm{E}[f(X)g(Y)] \overset{\epsilon \to 0}{=} \mathrm{E}[f(X)] \cdot \mathrm{E}[g(Y)]\ .$$

□

If we assume to have random variables that only take a finite number of values in $\mathbb{R}$, Theorem 3.24 can be improved to the following result.

Theorem 3.25 *For any two non-associated random variables X and Y taking only a finite number of values in $\mathbb{R}$ and any two functions f and g with finite $f(x)$ for any $x \in \mathbb{R}$ with $\Pr[X = x] > 0$ and finite $g(y)$ for any $y \in \mathbb{R}$ with $\Pr[Y = y] > 0$,*

$$\mathrm{E}[f(X)g(Y)] = \mathrm{E}[f(X)] \cdot \mathrm{E}[g(Y)] \; .$$

Proof. Since X and Y only take a finite number of values in $\mathbb{R}$, there are polynomials p and q such that $f(x) = p(x)$ for any $x \in \mathbb{R}$ with $\Pr[X = x] > 0$ and $g(y) = q(y)$ for any $y \in \mathbb{R}$ with $\Pr[Y = y] > 0$. Thus, we have

$$\mathrm{E}[f(X)g(Y)] = \mathrm{E}[p(X)q(Y)] = \mathrm{E}[p(X)] \cdot \mathrm{E}[q(Y)] = \mathrm{E}[f(X)] \cdot \mathrm{E}[g(Y)] \; .$$

□

Theorems 3.24 and 3.25 demonstrate that the expectation of products of functions of non-associated random variables can be treated in many cases as if the random variables were independent.

Tail estimates

Loéve stated the following result in [Loé77], which easily follows from the Markov inequality.

Theorem 3.26 (Loéve Inequality) *Let X be an arbitrary non-negative random variable and g be an arbitrary function that is non-negative and non-decreasing on $\mathbb{R}_+$. Then, for any $k \geq 0$ with $g(k) > 0$,*

$$\Pr[X \geq k] \leq \frac{\mathrm{E}[g(X)]}{g(k)} \; .$$

Proof. Since g is non-negative and non-decreasing on $\mathbb{R}_+$, it holds for every $k \geq 0$ that

$$\mathrm{E}[g(X)] = \sum_{x \in \mathbb{R}_+} g(x) \cdot \Pr[X = x] \geq g(k) \Pr[X \geq k] \; .$$

□

3.2.9 Basic Probability Distributions

In this section we introduce the most important probability distributions used in combination with discrete random variables. Many of the results presented in this section have been taken from [JK77]. Besides giving the expected value and the variance of these distributions, we will also state the factorial moments. Using these moments in Theorem 3.26 already yields very good tail estimates.

For instance, consider a random variable X with finite descending factorial moment $\mathrm{E}[X^{(r)}]$ for some $r \in \mathbb{N}$. Then, for any $k \in \mathbb{N}$ with $k \geq r$,

$$\Pr[X \geq k] \leq \frac{\mathrm{E}[X^{(r)}]}{k^{(r)}} \; .$$

Uniform distribution

A random variable X is called *uniformly distributed* over a finite set M of values in $\mathbb{R}$ if

$$\Pr[X = x] = \frac{1}{|M|}$$

for all $x \in M$. If $M = \{a, a+1, \ldots, b\}$ for some integers a, b with $a < b$, then

$$\mathrm{E}[X] = \frac{a+b}{2} \quad \text{and} \quad \mathrm{V}[X] = \frac{(b-a)^2}{12} + \frac{b-a}{6}$$

Since it is very easy to obtain sharp tail estimates in this case, we omit the definition of factorial moments.

Binomial distribution

A random variable X is called *binomially distributed* if there are parameters $p \in [0,1]$ and $n \in \mathbb{N}$ such that

$$\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$$

for all $k \in \{0, \ldots, n\}$. In this case,

$$\mathrm{E}[X] = n \cdot p \quad \text{and} \quad \mathrm{V}[X] = n \cdot p(1-p) \; .$$

Furthermore,

$$\mathrm{E}[X^{(r)}] = n^{(r)} p^r$$

for all $r \in \mathbb{N}$.

As an example, consider the situation that we have m balls in an urn, of which m_1 are red and m_2 are blue. We draw n balls at random out of this urn, one after the other, replacing each drawn ball. Let the random variable X denote the number of red balls drawn. Then X is binomially distributed with $p = m_1/m$.

Poisson distribution

A random variable X is called *Poisson distributed* if there is a parameter $\lambda > 0$ such that

$$\Pr[X = k] = \frac{\lambda^k}{k!} \cdot \mathrm{e}^{-\lambda}$$

for all $k \in \mathbb{N}_0$. In this case,

$$\mathrm{E}[X] = \lambda \quad \text{and} \quad \mathrm{V}[X] = \lambda \; .$$

Furthermore,

$$\mathrm{E}[X^{(r)}] = \lambda^r$$

for all $r \in \mathbb{N}$.

The binomial distribution converges towards the Poisson distribution if, in the example above, the number of balls in the urn, m, and the number of draws, n, is increased while keeping the number of red balls constant.

Geometric distribution

A random variable X is called *geometrically distributed* if there is a parameter $p \in [0, 1]$ such that

$$\Pr[X = k] = (1 - p)^{k-1} p$$

for all $k \in \mathbb{N}$. In this case,

$$\mathrm{E}[X] = \frac{1}{p} \quad \text{and} \quad \mathrm{V}[X] = \frac{1-p}{p^2} .$$

Furthermore,

$$\mathrm{E}[X^{[r]}] = \frac{r!}{p^r}$$

for all $r \in \mathbb{N}$.

As an example, suppose that we again have an urn with m balls, of which m_1 balls are red and m_2 balls are blue. We draw a ball at random from this urn and replace it until we draw a red ball. Let the random variable X denote the number of balls drawn. Then X is geometrically distributed with $p = m_1/m$.

Hypergeometric distribution

A random variable X is called *hypergeometrically distributed* if there are parameters N, M, and n such that

$$\Pr[X = k] = \frac{\binom{M}{k}\binom{N-M}{n-k}}{\binom{N}{n}}$$

for all $k \in \{0, \ldots, n\}$. In this case, we obtain for $p = M/N$

$$\mathrm{E}[X] = n \cdot p \quad \text{and} \quad \mathrm{V}[X] = \frac{N-n}{N-1} \cdot n \cdot p(1-p) .$$

Furthermore,

$$\mathrm{E}[X^{(r)}] = \frac{n^{(r)} \cdot M^{(r)}}{N^{(r)}}$$

for all $r \in \mathbb{N}$.

As an example, consider the situation that we have N balls in an urn, of which M are red and $N - M$ are blue. We draw n balls at random out of this urn, one after the other, *without* replacing each drawn ball. Let the random variable X denote the number of red balls drawn. Then X is hypergeometrically distributed.

Negative binomial distribution

A random variable X is called *negative binomially distributed* if there are parameters $p \in (0, 1]$ and $j \in \mathbb{N}$ such that

$$\Pr[X = k] = \binom{k-1}{j-1} p^j (1-p)^{k-j}$$

for all $k \in \mathbb{N}$ with $k \geq j$. Instead of using p and $1 - p$, the negative binomial distribution is also sometimes expressed by the use of $Q = 1/p$ and $P = (1 - p)/p$:

$$\Pr[X = k] = \binom{k-1}{j-1} P^{-j} \left(\frac{P}{Q}\right)^k .$$

For this distribution we obtain

$$\mathrm{E}[X] = Q \quad \text{and} \quad \mathrm{V}[X] = j \cdot Q(1 - Q) \ .$$

Furthermore,

$$\mathrm{E}[X^{[r]}] = j^{[r]} \cdot Q^r$$

for all $r \in \mathbb{N}$. For $j = 1$ the negative binomial distribution is equal to the geometric distribution with parameter p.

As an example, consider the situation that we have an urn with m balls, of which m_1 balls are red and m_2 balls are blue. We draw one ball after the other at random from this urn and replace it until we have drawn altogether j red balls. Let the random variable X denote the number of balls drawn. Then X is negative binomially distributed with $p = m_1/m$.

Negative hypergeometric distribution

A random variable X is called *negative hypergeometrically distributed* if there are parameters k, j, N, and M such that

$$\Pr[X = k] = \binom{k-1}{j-1} \frac{M^{(j)}(N - M)^{(k-j)}}{N^{(k)}} \ .$$

for all $k \in \mathbb{N}$ with $k \geq j$. For this distribution we obtain

$$\mathrm{E}[X] = \frac{j \cdot (N+1)}{M+1} \quad \text{and} \quad \mathrm{V}[X] = \frac{j(N+1)(N-M)(M+1-j)}{(M+1)^2(M+2)} \ .$$

Furthermore,

$$\mathrm{E}[X^{[r]}] = \frac{j^{[r]}(N+1)^{[r]}}{(M+1)^{[r]}} \ .$$

for all $r \in \mathbb{N}$.

As an example, consider the situation that we have an urn with N balls, of which M balls are red and $N - M$ balls are blue. We draw one ball after the other at random from this urn *without* replacing them until we have drawn altogether j red balls. Let the random variable X denote the number of balls drawn. Then X is negative hypergeometrically distributed.

3.2.10 Approximations of Probability Distributions

In this section we demonstrate that in certain cases we can approximate probability distributions via standard distributions such as the Poisson or Gaussian probability distribution.

Approximating the binomial distribution via the Poisson distribution

Let $X_1, \ldots, X_n$ be independent Bernoulli random variables with probabilities $p_i = \Pr[X_i = 1]$ for all $1 \leq i \leq n$. We define $S_n = \sum_{i=1}^{n} X_i$. Let $B = (b_0, b_1, \ldots, b_n)$ be the probability distribution of the sum S_n, where $b_k = \Pr[S_n = k]$. Furthermore, let $\Pi = (\pi_0, \pi_1, \ldots)$ be the Poisson distribution with parameter λ, where

$$\pi_k = \frac{\lambda^k}{k!} \cdot \mathrm{e}^{-\lambda} \ .$$

Let $||B - \Pi||$ be defined as $\sum_{k=0}^{\infty} |b_k - \pi_k|$. Then the following theorem holds.

Theorem 3.27 (Presman [Pre85]) *For $\lambda = \mathrm{E}[S_n]$ it holds*

$$||B - \Pi|| \leq 2 \sum_{i=1}^{n} p_i^2 \ .$$

Since $\sum_{i=1}^{n} p_i^2 \leq \lambda \max_{i \in [n]} p_i$, this also implies that $||B - \Pi|| \leq 2\lambda \max_{i \in [n]} p_i$.

Approximations using the normal distribution

The *normal* or *Gaussian* distribution is defined by the probability density function

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \cdot \mathrm{e}^{-x^2/2} \ .$$

The corresponding distribution function is defined by

$$\Phi(x) = \int_{-\infty}^{x} \phi(u) du = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \mathrm{e}^{-u^2/2} du \ .$$

X is called a *Gaussian random variable* if $\Pr[X \leq x] = \Phi(x)$ for all $x \in \mathrm{IR}$.

Now, consider the function

$$\Theta(x) = \mathrm{e}^{x^2/2} \int_{x}^{\infty} \phi(u) du \ .$$

Thus $\Theta(0) = 1/2$, and it is not difficult to see that Θ decreases for $x \geq 0$. Furthermore, it has been shown in [IM74] p. 17 and [JK82] p. 505 that

$$\frac{2}{\sqrt{2\pi}(x + \sqrt{x^2 + 4})} \leq \Theta(x) \leq \frac{4}{\sqrt{2\pi}(3x + \sqrt{x^2 + 8})} \ .$$

This implies the following result.

Theorem 3.28 *If X is a Gaussian random variable, then*

$$\frac{2}{\sqrt{2\pi}(x + \sqrt{x^2 + 4})} \cdot \mathrm{e}^{-x^2/2} \leq \Pr[X \geq x] \leq \frac{4}{\sqrt{2\pi}(3x + \sqrt{x^2 + 8})} \cdot \mathrm{e}^{-x^2/2}$$

for any $x \geq 0$. Or, in a more simple way,

$$\frac{1}{\sqrt{2\pi}(1 + x)} \cdot \mathrm{e}^{-x^2/2} \leq \Pr[X \geq x] \leq \frac{1}{\sqrt{\pi}(1 + x)} \cdot \mathrm{e}^{-x^2/2} \ .$$

As stated in the following well-known result, in the infinite case, this inequality holds also for sums of random variables with any other common distribution.

Theorem 3.29 (Central Limit Theorem) *Let $S_n = \sum_{i=1}^{n} X_i$ be the sum of n discrete independent random variables with common distribution having finite expected value μ and variance ν. Then, for any $a < b$,*

$$\lim_{n \to \infty} \Pr\left[a < \frac{S_n - n\mu}{\sqrt{n \cdot \nu}} < b\right] = \Phi(b) - \Phi(a) \ .$$

Combining Theorem 3.29 with Theorem 3.28, we obtain the following result.

Corollary 3.30 *Let $S_n = \sum_{i=1}^n X_i$ be the sum of n discrete independent random variables with common distribution having finite expected value μ and variance ν. Then, for $n \to \infty$, it holds for any $x \geq 0$ that*

$$\frac{1}{\sqrt{2\pi}(1+x)} \cdot \mathrm{e}^{-x^2/2} \leq \Pr\left[\frac{S_n - n\mu}{\sqrt{n \cdot \nu}} \geq x\right] \leq \frac{1}{\sqrt{\pi}(1+x)} \cdot \mathrm{e}^{-x^2/2}$$

The drawback of the Central Limit Theorem is that it does not say anything about the rate of convergence towards the Gaussian distribution. We will quantify this rate of convergence for the case that S_n is normalized to

$$S_n = \frac{\sum_{i=1}^n X_i}{\sqrt{n \cdot \nu}}$$

where $X_1, \ldots, X_n$ is a sequence of independent, identically distributed random variables with $\mathrm{E}[X_i] = 0$, $\mathrm{V}[X_i] = \nu$, and $\mathrm{E}[|X_1|^3] = \gamma < \infty$. For all $x \in \mathbb{R}$, let $F_n(x) = \Pr[S_n \leq x]$. Then the following theorem holds (for a proof, see e.g., [GK68]).

Theorem 3.31 (Berry and Esseen) *For every $n \in \mathbb{N}$ we have*

$$\sup_x |F_n(x) - \Phi(x)| \leq \frac{\gamma}{\sqrt{n \cdot \nu^3}} \cdot \frac{1}{(1+|x|)^3} \ .$$

The bound in this theorem is sharp up to some constant $(2\pi)^{-1/2}$ [Shi96].

Convergence bounds have also been shown for non-identically distributed random variables. Let $X_1, \ldots, X_n$ be independent binary random variables and let

$$S = \sum_{i=1}^n X_i, \quad \mu = \mathrm{E}[S] = \sum_{i=1}^n \mathrm{E}[X_i], \quad \sigma^2 = \mathrm{V}[S], \quad \text{and } \gamma = \mathrm{E}[(S-\mu)^3] \ .$$

Furthermore, let

$$\Gamma(x) = \Phi(x) + \frac{\gamma}{6\sigma^3}(1-x^2)\phi(x) \ .$$

Then the following theorem holds.

Theorem 3.32 (Mikhailov [Mik93]) *Suppose that $\sigma \geq 10$. Then, for every $0 \leq m \leq n$,*

$$\left| \Pr[S \leq m] - \Gamma\left(\frac{m - \mu + 1/2}{\sigma}\right) \right| \leq \frac{\sigma + 3}{4\sigma^3} \ .$$

As noted by Mikhailov, particular examples demonstrate that the theorem also holds in the range $0 < \sigma \leq 10$. The restriction $\sigma \geq 10$ is only a technical problem. For the range $5 \leq \sigma \leq 10$, the same proof yields a rougher bound of $(2\sigma + 5)/(5\sigma^3)$.

3.3 Tail Estimates

In the following we present tail estimates for sums of random variables that are in many cases much better than the tail estimates provided by the theorems in Section 3.2.10, and that apply to much more cases than studied in Section 3.2.9. We will always assume that $X_1, \ldots, X_n$ are discrete random variables with finite first and second moments. We define

$$S_n = \sum_{i=1}^n X_i$$

to be the sum of these variables and

$$A_n = \frac{S_n}{n}$$

to be the average. If $\mathrm{E}[X_i] = \mu_i$ and $\mathrm{V}[X_i] = \nu_i$ for all $i \in [n]$, then

$$\mathrm{E}[S_n] = \sum_{i=1}^{n} \mu_i \quad \text{and} \quad \mathrm{V}[S_n] = \sum_{i=1}^{n} \nu_i \ .$$

Furthermore,

$$\mathrm{E}[A_n] = \mathrm{E}[S_n]/n \quad \text{and} \quad \mathrm{V}[A_n] = \mathrm{V}[S_n]/n^2 \ .$$

We often use $\mu = \mathrm{E}[S_n]$, $\nu = \mathrm{V}[S_n]$, $\bar{\mu} = (\mu_1 + \ldots + \mu_n)/n$, and $\bar{\nu} = (\nu_1 + \ldots + \nu_n)/n$.

Let us first review the concepts of majorization and Schur-convexity. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be vectors in IR^n, and let $x_{[i]}$ and $y_{[i]}$ denote the ith largest component of $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively. Then we say that $\boldsymbol{x} \prec \boldsymbol{y}$ ($\boldsymbol{x}$ is *majorized* by $\boldsymbol{y}$) if

$$\sum_{i=1}^{k} x_{[i]} \leq \sum_{i=1}^{k} y_{[i]} \quad \text{for all } 1 \leq k \leq n \text{ and } \quad \sum_{i=1}^{n} x_{[i]} = \sum_{i=1}^{n} y_{[i]} \ .$$

To give an intuition for this definition, $\boldsymbol{x}$ is majorized by $\boldsymbol{y}$ if its entries are, in a certain sense, more balanced than the entries in $\boldsymbol{y}$. (Note that the symbol $\prec$ has already been used for stochastic domination. Since for both notions $\prec$ is common in the literature, we avoided using a different symbol for one of them. To ensure a unique meaning of $\prec$, it will always mean majorization in the context of vectors and dominance in the context of random variables.)

A function f of n arguments is *Schur-convex* on a set $A \subseteq \mathrm{IR}^n$ if, for all $\boldsymbol{x}, \boldsymbol{y} \in A$,

$$\boldsymbol{x} \prec \boldsymbol{y} \quad \Rightarrow f(\boldsymbol{x}) \leq f(\boldsymbol{y}) \ .$$

A function f is *Schur-concave* if

$$\boldsymbol{x} \prec \boldsymbol{y} \quad \Rightarrow f(\boldsymbol{x}) \geq f(\boldsymbol{y}) \ .$$

Since $\prec$ is transitive (i.e., if $\boldsymbol{x} \prec \boldsymbol{y}$ and $\boldsymbol{y} \prec \boldsymbol{z}$, then $\boldsymbol{x} \prec \boldsymbol{z}$), it is often quite easy to prove that a function f is Schur-convex or Schur-concave: all that has to be shown is that for every vector, all of its direct successors or predecessors concerning $\prec$ fulfill the respective Schur condition.

The following fact follows immediately from the definition of Schur-convex and Schur-concave functions.

Fact 3.33 *Assume that we have a set A of vectors with the property that for some $\boldsymbol{x} \in A$, $\boldsymbol{x} \prec \boldsymbol{y}$ for all $\boldsymbol{y} \in A$. If f is Schur-convex on A, then $f(\boldsymbol{x}) \leq f(\boldsymbol{y})$ for all $\boldsymbol{y} \in A$. If f is Schur-concave, the inequality holds with $\geq$ in place of $\leq$.*

Furthermore, assume that we have a set B of vectors with the property that for some $\boldsymbol{x} \in B$, $\boldsymbol{x} \succ \boldsymbol{y}$ for all $\boldsymbol{y} \in B$. If f is Schur-convex on A, then $f(\boldsymbol{x}) \geq f(\boldsymbol{y})$ for all $\boldsymbol{y} \in B$. If f is Schur-concave, the inequality holds with $\leq$ in place of $\geq$.

The fact immediately implies the following important result. (We call a set $A \subseteq \mathrm{IR}^n$ *convex* w.r.t. $\succ$ if all $\boldsymbol{x} \in A$ fulfill $\sum_i x_i = m$ for some fixed $m \in \mathrm{IR}$ and for every $\boldsymbol{x}$ in A it holds that all $\boldsymbol{y} \in \mathrm{IR}^n$ with $\boldsymbol{y} \prec \boldsymbol{x}$ are also in A.)

Corollary 3.34 *Let f be a Schur-convex function on a convex set $A \subseteq \mathbb{R}^n$. Then for any $\boldsymbol{x} = (x_1, \ldots, x_n) \in A$,*

$$f(x_1, \ldots, x_n) \geq f(m, \ldots, m) \quad \text{where } m = \frac{x_1 + \ldots + x_n}{n} .$$

For a Schur-concave function f, the inequality holds with $\leq$ in place of $\geq$.

Corollary 3.34 will be useful when dealing with sums of non-identically distributed random variables.

3.3.1 Tail Estimates Based on the Expectation

First, we study sums of independent binary random variables. Tail estimates for these sums have among many others been investigated by Chernoff [Che52] and are often called *Chernoff bounds*. It has been found convenient to use exponential moments of random variables to derive these tail estimates. The idea of using exponential moments was apparently first used by S. N. Bernstein. His method had a significant influence on deriving tail estimates also for many other problems. We present here a proof given by Hagerup and Rüb [HR90]. Bound (3.20) was taken from [McD98].

Theorem 3.35 (Chernoff Bound) *If $X_1, \ldots, X_n$ are independent binary random variables, then it holds for all $\delta \geq 0$ that*

$$\begin{aligned} \Pr[S_n \geq (1+\delta)\mu] &\leq \left(\frac{\mathrm{e}^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} && (3.19)\\ &\leq \mathrm{e}^{-\frac{\delta^2 \mu}{2(1+\delta/3)}} && (3.20)\\ &\leq \mathrm{e}^{-\min[\delta^2,\, \delta]\cdot\mu/3} . \end{aligned}$$

Furthermore, it holds for all $0 \leq \delta \leq 1$ that

$$\begin{aligned} \Pr[S_n \leq (1-\delta)\mu] &\leq \left(\frac{\mathrm{e}^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu} && (3.21)\\ &\leq \mathrm{e}^{-\delta^2\mu/2} . \end{aligned}$$

Proof. We will only show (3.19) and (3.21). First we present a proof for inequality (3.19). For all $i \in [n]$, let $p_i = \mathrm{E}[X_i]$. According to the Loéve inequality (see Theorem 3.26) we obtain that, for any function $g(x) = e^{h \cdot x}$ with $h > 0$ and any $\delta \geq 0$,

$$\Pr[S_n \geq (1+\delta)\mu] \leq \mathrm{e}^{-h(1+\delta)\mu} \cdot \mathrm{E}[\mathrm{e}^{h \cdot S_n}] . \tag{3.22}$$

Since $X_1, \ldots, X_n$ are independent, it follows from Claim 3.10 that

$$\begin{aligned} \mathrm{E}[\mathrm{e}^{h\cdot S_n}] &= \mathrm{E}[\mathrm{e}^{h(X_1+\ldots+X_n)}] = \mathrm{E}[\mathrm{e}^{h\cdot X_1} \cdots \mathrm{e}^{h\cdot X_n}] = \textstyle\prod_{i=1}^n \mathrm{E}[\mathrm{e}^{h\cdot X_i}]\\ &= \prod_{i=1}^n (p_i \mathrm{e}^h + (1-p_i)) = \textstyle\prod_{i=1}^n (1 + p_i(\mathrm{e}^h - 1))\\ &\leq \prod_{i=1}^n \mathrm{e}^{p_i(\mathrm{e}^h-1)} \quad \text{since } 1+x \leq \mathrm{e}^x \text{ for all } x\\ &= \mathrm{e}^{\mu(\mathrm{e}^h-1)} . \end{aligned}$$

This yields together with inequality (3.22) that

$$\Pr[S_n \geq (1+\delta)\mu] \leq \mathrm{e}^{-h(1+\delta)\mu} \cdot \mathrm{e}^{\mu(\mathrm{e}^h - 1)} = \mathrm{e}^{-(1+h(1+\delta)-\mathrm{e}^h)\mu} . \tag{3.23}$$

The right-hand side of (3.23) attains its minimum at $h = h_0$, where $h_0 = \ln(1+\delta)$. Inserting this in (3.23) yields

$$\Pr[S_n \geq (1+\delta)\mu] \leq (1+\delta)^{-(1+\delta)\mu} \cdot \mathrm{e}^{\delta\cdot\mu} = \left(\frac{\mathrm{e}^\delta}{(1+\delta)^{1+\delta}}\right)^\mu .$$

This completes the proof for inequality (3.19). The proof of (3.21) is similar and can be found in Section 3.3.4. □

If $X_1, \ldots, X_n$ are identically distributed binary random variables, the bounds in Theorem 3.35 can be easily improved:

Suppose that $\mathrm{E}[X_i] = p$ for all $i \in [n]$. Let $0 < t < 1$, $t \geq p$, and $h \geq 0$. Choosing δ to make $(1+\delta)\mu = t \cdot n$, we get from the calculations in the proof of Theorem 3.35 that

$$\Pr[S_n \geq t \cdot n] \leq \mathrm{e}^{-h\cdot t\cdot n}(p \cdot \mathrm{e}^h + (1-p))^n .$$

For $h = \ln(t(1-p)/[p(1-t)])$ this becomes

$$\Pr[S_n \geq t \cdot n] \leq \left(\frac{p(1-t)}{t(1-p)}\right)^{t\cdot n} \left(\frac{t(1-p)}{1-t} + (1-p)\right)^n = \left(\frac{p(1-t)}{t(1-p)}\right)^{t\cdot n} \left(\frac{1-p}{1-t}\right)^n ,$$

from which we obtain

$$\Pr[S_n \geq t \cdot n] \leq \left[\left(\frac{p}{t}\right)^t \left(\frac{1-p}{1-t}\right)^{1-t}\right]^n . \tag{3.24}$$

It is not difficult to show that this bound is better than the bounds in Theorem 3.35. Therefore, one may ask, whether this bound also holds for non-identically distributed binary random variables. In fact, Gleser [Gle75] showed that if $X_1, \ldots, X_n$ are independent binary random variables with $\Pr[X_i = 1] = p_i$ for all $i \in [n]$, then the function

$$F(p_1, \ldots, p_n, t) = \Pr[X_1 + \ldots + X_n \leq t]$$

is Schur-concave in $(p_1, \ldots, p_n)$ for $0 \leq t \leq n\bar{p} - 2$ and Schur-convex for $n\bar{p} + 1 \leq t \leq n$, where $\bar{p} = (p_1 + \ldots + p_n)/n$. Using this property together with Corollary 3.34 implies that the tail estimate (3.24) also holds for non-identically distributed binary random variables. This will be shown explicitly in the proof of the next theorem.

Bounded random variables

The next theorem is due to Hoeffding [Hoe63]. Therefore, it is usually called *Chernoff-Hoeffding bound* or *Hoeffding bound.*

Theorem 3.36 (Hoeffding Bound) *If $X_1, \ldots, X_n$ are independent and $0 \leq X_i \leq 1$ for all $i \in [n]$, then for $\bar{\mu} < t < 1$*

$$\begin{aligned} \Pr[A_n \geq t] &\leq \left[\left(\frac{\bar{\mu}}{t}\right)^t \left(\frac{1-\bar{\mu}}{1-t}\right)^{1-t}\right]^n && (3.25)\\ &\leq \mathrm{e}^{-nd^2 g(\bar{\mu})} && (3.26)\\ &\leq \mathrm{e}^{-2nd^2} , && (3.27) \end{aligned}$$

where $d = t - \bar{\mu}$ *and*

$$g(x) = \begin{cases} \frac{1}{1-2x} \ln \frac{1-x}{x} & : \quad 0 < x < \frac{1}{2} \\ \frac{1}{2x(1-x)} & : \quad \frac{1}{2} \leq x < 1 \end{cases} .$$

The proof of Theorem 3.36 can be found in Section 3.3.4. It is easy to extend the theorem to arbitrary sequences of random variables with $a \leq X_i \leq b$ by simply shifting the X_i's to a range of $[0, b-a]$ and then scaling them to $[0, 1]$. In this case, the values $\bar{\mu}$, t, and d in the three upper bounds of the theorem have to be replaced by $(\bar{\mu} - a)/(b-a)$, $(t-a)/(b-a)$, and $d/(b-a)$, respectively. As noted by Hoeffding [Hoe63], the bound in (3.25) is the best that can be obtained from the exponential moment method (i.e., using the Loéve Inequality with $g(x) = \mathrm{e}^{hx}$). Naveau [Nav97] and Philips and Nelson [PN95] showed that there are functions that yield better tail estimates (we will come back to this point in the proof of Theorem 3.46).

If the ranges of the random variables differ significantly, then it might be more useful to apply the following theorem, which is also due to Hoeffding. Its proof can be found, e.g., in [Hoe63, McD98].

Theorem 3.37 ([Hoe63]) *If* $X_1, \ldots, X_n$ *are independent and* $a_i \leq X_i \leq b_i$ *for all* $i \in [n]$*, then for* $d > 0$

$$\Pr[A_n - \bar{\mu} \geq d] \leq \mathrm{e}^{-2n^2 d^2/(\sum_{i=1}^n (b_i - a_i)^2)} .$$

Unbounded random variables

If some of the X_i's are unbounded, then the previous theorems cannot be applied. We will show that for the case that the X_i's are independent geometrically distributed random variables, good tail estimates can nevertheless be shown. Recall that a random variable is geometrically distributed if there is a $0 < p < 1$ with $\Pr[X = k] = (1-p)^{k-1} p$ for all $k \geq 1$. See Section 3.3.4 for the proof of the following result.

Theorem 3.38 *Let* $X_1, \ldots, X_n$ *are independent, geometrically distributed random variables with parameters* $p_1, \ldots, p_n \in (0, 1)$*, and let* $p = \min_i p_i$*. Then*

- *for all* $\delta > 0$ *it holds that*

$$\Pr[S_n \geq (1+\delta)\mu] \leq \left(1 + \frac{\delta \mu p}{n}\right)^n \mathrm{e}^{-\delta \mu p} , \tag{3.28}$$

- *and for all* $0 \leq \delta \leq 1$*, it holds that*

$$\Pr[S_n \leq (1-\delta)\mu] \leq \mathrm{e}^{-\delta^2 \mu p/2} . \tag{3.29}$$

This theorem can be used, for instance, to prove much better tail estimates for the Coupon Collector Problem than presented in Example 2 of Section 3.2.7.

The following theorem due to Merkle and Petrovic [MP97] offers alternative ways of proving tail estimates for geometrical — and, in general, negative binomial — random variables.

Theorem 3.39 ([MP97]) *Let* $X_1, \ldots, X_n$ *be independent geometric random variables with parameters* $p_1, \ldots, p_n$*. That is,*

$$\Pr[X_i = k] = (1 - p_i)^{k-1} p_i$$

for all $k \geq 1$. Then the function

$$F(p_1, \ldots, p_n, t) = \Pr[X_1 + \ldots + X_n \leq t]$$

is Schur-concave with respect to $(p_1, \ldots, p_n)$ for every real t.

Theorem 3.39 and Fact 3.33 together imply that $\Pr[S_n \geq t]$ is maximal if as many of the X_i as possible have a parameter p_i that is equal to $p = \min_i p_i$. Furthermore, $\Pr[S_n \leq t]$ is maximal if all of the X_i have the same probability parameter. This allows to bound $\Pr[S_n \leq t]$ in terms of $\bar{p} = (p_1 + \ldots + p_n)/n$ instead of using p as in Theorem 3.38. However, note that changing the parameters $p_1, \ldots, p_n$ changes $\mathrm{E}[S_n]$, even if their sum $p_1 + \ldots + p_n$ remains fixed.

Using bounds for the expectation

Sometimes it may not be possible to calculate $\mathrm{E}[S_n]$ exactly. In this case, it may be more convenient to use upper or lower bounds for $\mathrm{E}[S_n]$. We will show that in all bounds presented above, $\mathrm{E}[S_n]$ can be replaced by upper or lower bounds on $\mathrm{E}[S_n]$ (that do not violate the restrictions on the X_i in S_n), depending on whether we need estimates for $\Pr[S_n \geq t]$ or for $\Pr[S_n \leq t]$. For this we need the following claim. (Recall that here "$\succ$" means stochastic domination.)

Claim 3.40 *If X_1, X_2, Y_1, and Y_2 are independent random variables with $X_1 \succ Y_1$ and $X_2 \succ Y_2$, then $(X_1 + X_2) \succ (Y_1 + Y_2)$.*

Proof. For all $z \in \mathrm{IR}$ it holds:

$$\begin{aligned}
\Pr[X_1 + X_2 \geq z] &= \sum_{z'} \Pr[X_1 = z'] \cdot \Pr[X_2 \geq z - z'] \\
&\geq \sum_{z'} \Pr[X_1 = z'] \cdot \Pr[Y_2 \geq z - z'] \\
&= \Pr[X_1 + Y_2 \geq z] = \sum_{z'} \Pr[Y_2 = z'] \cdot \Pr[X_1 \geq z - z'] \\
&\geq \sum_{z'} \Pr[Y_2 = z'] \cdot \Pr[Y_1 \geq z - z'] = \Pr[Y_1 + Y_2 \geq z] \; .
\end{aligned}$$

□

This claim immediately yields the following result.

Lemma 3.41 *For any two sets of independent random variables $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_n$ it holds:*

- *if $X_i \succ Y_i$ for all $i \in [n]$, then*

  $$\Pr\left[\sum_{i=1}^{n} X_i \leq t\right] \leq \Pr\left[\sum_{i=1}^{n} Y_i \leq t\right]$$

  for every $t \in \mathrm{IR}$, and

- *if $X_i \prec Y_i$ for all $i \in [n]$, then*

  $$\Pr\left[\sum_{i=1}^{n} X_i \geq t\right] \leq \Pr\left[\sum_{i=1}^{n} Y_i \geq t\right]$$

  for every $t \in \mathrm{IR}$.

Clearly, for any of the cases studied in Section 3.3.1 it holds that for any sum S_n of random variables with $\mathrm{E}[S_n] \leq \mu$, where μ does not violate the restrictions on the random variables in S_n, a sum S_n' of random variables with the same restrictions that dominate the variables in S_n can be defined with $\mathrm{E}[S_n'] = \mu$. Furthermore, for any sum S_n of random variables with $\mathrm{E}[S_n] \geq \mu$, where μ does not violate the restrictions on the random variables in S_n, a sum S_n' of random variables with the same restrictions that are dominated by the variables in S_n can be defined with $\mathrm{E}[S_n'] = \mu$. Thus, Lemma 3.41 implies that we can replace μ in the bounds for $\Pr[S_n \geq (1+\delta)\mu]$ above by any $\mu \geq \mathrm{E}[X]$, and we can replace μ in the bounds for $\Pr[S_n \leq (1-\delta)\mu]$ by any $\mu \leq \mathrm{E}[X]$, subject to the condition that μ does not violate the restrictions on the random variables in S_n.

Tail estimates for products of random variables

Suppose that we want to bound the probability that the product of independent, non-negative random variables $X_1, \ldots, X_n$ is at most or at least t. In this case we simply set $Y_i = \ln X_i$ for all $i \in [n]$. Then $\Pr[\prod_i X_i \geq t]$ is equivalent to $\Pr[\sum_i Y_i \geq \ln t]$, and $\Pr[\prod_i X_i \leq t]$ is equivalent to $\Pr[\sum_i Y_i \leq \ln t]$. Thus we can apply the tail estimates derived above for sums of independent random variables.

3.3.2 Tail Estimates Based on the Variance

Now we present tail estimates that use the variances of the involved random variables. We start with a result (not far away from the "ideal" bound in Corollary 3.30) that has a simple proof.

Theorem 3.42 ([DP98]) *If $X_1, \ldots, X_n$ are independent random variables with $X_i - \mathrm{E}[X_i] \leq 1$ and $\mathrm{V}[\sum_i X_i] = \nu$, then for any $0 \leq d \leq 2\nu$,*

$$\Pr[S_n \geq \mu + d] \leq \mathrm{e}^{-d^2/4\nu} .$$

Proof. For all $i \in [n]$, let $\mu_i = \mathrm{E}[X_i]$ and $\nu_i = \mathrm{V}[X_i]$. It holds for every $h > 0$ that

$$\begin{aligned} \Pr[S_n \geq \mu + d] &= \Pr[\sum_i (X_i - \mu_i) \geq d] = \Pr[\mathrm{e}^{h\sum_i (X_i-\mu_i)} \geq \mathrm{e}^{h \cdot d}] \\ &\leq \mathrm{E}[\mathrm{e}^{h\sum_i (X_i-\mu_i)}]/\mathrm{e}^{h\cdot d} . \end{aligned}$$

It is easy to check that $\mathrm{e}^x \leq 1 + x + x^2$ for every $x \leq 1$ and $\mathrm{e}^x \geq 1 + x$ for every $x \in \mathbb{R}$. If $h(X_i - \mu_i) \leq 1$ for every $i \in [n]$, we have

$$\begin{aligned} \mathrm{E}[\mathrm{e}^{h\sum_i (X_i-\mu_i)}] &= \prod_i \mathrm{E}[\mathrm{e}^{h(X_i-\mu_i)}] \\ &\leq \prod_i \mathrm{E}[1 + h(X_i-\mu_i) + h^2(X_i-\mu_i)^2] \\ &= \prod_i (1 + h^2\nu_i) \leq \prod_i \mathrm{e}^{h^2\nu_i} = \mathrm{e}^{h^2\nu} . \end{aligned}$$

Thus,

$$\Pr[S_n \geq \mu + d] \leq \mathrm{e}^{h^2\nu}/\mathrm{e}^{h\cdot d} . \tag{3.30}$$

This term is minimized for $h = h_0$ with $h_0 = d/(2\nu)$. Using this in inequality (3.30) yields the theorem. □

Theorem 3.42 yields better tail estimates than the above theorems if ν is much smaller than μ. (This is, for instance, the case for Example 1 in Section 3.2.7 if $m \gg n$.) Even better bounds are given in the following theorem (recall the definition of $\bar{\nu}$ at the beginning of Section 3.3). Its proof can be found in Section 3.3.4.

Theorem 3.43 ([Hoe63]) *If $X_1, \ldots, X_n$ are independent, and $\mathrm{E}[X_i] = 0$ and $X_i \leq b$ for all $i \in [n]$, then for any $0 < t < b$,*

$$\Pr[A_n \geq t] \leq \left[\left(1 + \frac{bt}{\bar{\nu}}\right)^{-(1+\frac{bt}{\bar{\nu}})\frac{\bar{\nu}}{b^2+\bar{\nu}}} \left(1 - \frac{t}{b}\right)^{-(1-\frac{t}{b})\frac{b^2}{b^2+\bar{\nu}}} \right]^n \tag{3.31}$$

$$\leq \mathrm{e}^{-\frac{nt}{b}\left[(1+\frac{\bar{\nu}}{bt})\ln(1+\frac{bt}{\bar{\nu}})-1\right]} . \tag{3.32}$$

Note that this theorem can be applied to any collection of upper-bounded random variables $X_1, \ldots, X_n$ by using shifts and scaling. As noticed by Hoeffding [Hoe63], inequality (3.31) is the best that can be obtained from using the exponential moment method.

McDiarmid [McD98] simplified the bounds in Theorem 3.43 to obtain bounds that look very similar to the Chernoff bounds (see Theorem 3.35).

Theorem 3.44 ([McD98]) *If $X_1, \ldots, X_n$ are independent, and $X_i - \mathrm{E}[X_i] \leq b$ for all $i \in [n]$, then for any $t \geq 0$,*

$$\Pr[S_n - \mu \geq d] \leq \left(\frac{\mathrm{e}^{\delta}}{(1+\delta)^{1+\delta}} \right)^{\nu/b^2}$$

$$\leq \mathrm{e}^{-\frac{d^2}{2\nu(1+\delta/3)}} \tag{3.33}$$

where $\delta = b \cdot d/\nu$.

Comparing (3.33) with Corollary 3.30, it becomes clear why Hoeffding's bounds are essentially best possible. What is left is essentially the factor in front of the e-Formula in Corollary 3.30. Talagrand presents bounds that also include an approximation for this factor [Tal95b]. His bounds are the best obtained so far.

Using bounds for the variance

As for the expected value $\mathrm{E}[S_n]$, it may be sometimes not possible to calculate $\mathrm{V}[S_n]$ exactly. In this case it may be more convenient to use an upper bound for $\mathrm{V}[S_n]$. Clearly, for the theorems presented above it holds that the probability bounds are monotonically decreasing with an increasing $\mathrm{V}[S_n]$. Hence, upper bounds cause no problems.

3.3.3 Tail Estimates for Dependent Random Variables

Clearly, all bounds presented above also hold for collections of mutually non-associated random variables, since all of them are based on Bernstein's exponential moment method and $\mathrm{E}[\mathrm{e}^{hX}]$ can be expressed as $\sum_{i=0}^{\infty} h^i \mathrm{E}[X^i]/i!$. In the following we will study the case of k-wise independent or k-wise non-associated random variables and of k-wise uncorrelated random variables. We will also consider some more general types of dependent random variables.

k-wise independent random variables

First, we consider the case that the random variables are k-wise independent. Recall that $\cosh(x) = (\mathrm{e}^x + \mathrm{e}^{-x})/2$. The following theorem is due to Schmitt, Siegel, and Srinivasan.

Theorem 3.45 ([SSS95]) *If $X_1, \ldots, X_n$ are k-wise independent random variables that satisfy $|X_i - \mathrm{E}[X_i]| \leq 1$, then for any even k and any $d > 0$,*

$$\Pr[|S_n - \mu| \geq d] \leq \sqrt{2}\cosh\left(\sqrt{\frac{k^3}{36\nu}}\right) \cdot \left(\frac{k \cdot \nu}{\mathrm{e}\, d^2}\right)^{k/2} .$$

For non-negative random variables, the following theorem was shown by the same authors. Its proof can be found in Section 3.3.4.

Theorem 3.46 ([SSS95]) *If S is the sum of k-wise independent random variables, each of which is confined to the interval $[0, 1]$, then we have:*

(1) For $\delta \leq 1$,

   (a) if $k \leq \lfloor \delta^2\mu \cdot \mathrm{e}^{-1/3} \rfloor$, then $\Pr[|S - \mu| \geq \delta\mu] \leq \mathrm{e}^{-\lfloor k/2 \rfloor}$, and

   (b) if $k = \lfloor \delta^2\mu \cdot \mathrm{e}^{-1/3} \rfloor$, then $\Pr[|S - \mu| \geq \delta\mu] \leq \mathrm{e}^{-\lfloor \delta^2\mu/3 \rfloor}$.

(2) For $\delta \geq 1$,

   (a) if $k \leq \lfloor \delta\mu \cdot \mathrm{e}^{-1/3} \rfloor$, then $\Pr[|S - \mu| \geq \delta\mu] \leq \mathrm{e}^{-\lfloor k/2 \rfloor}$, and

   (b) if $k = \lfloor \delta\mu \cdot \mathrm{e}^{-1/3} \rfloor$, then $\Pr[|S - \mu| \geq \delta\mu] \leq \mathrm{e}^{-\lfloor \delta\mu/3 \rfloor}$.

(3) If $\delta \geq 1$ and $k = \lceil \delta\mu \rceil$,

$$\Pr[|S - \mu| \geq \delta\mu] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu .$$

k-wise non-associated and k-wise uncorrelated random variables

In the proof of Theorem 3.46 in Section 3.3.4 solely multilinear polynomials of the form

$$P_j(X_1, \ldots, X_n) = \sum_{i_1 < i_2 < \ldots < i_j} X_{i_1} X_{i_2} \ldots X_{i_j}$$

are used. Since their expected values are the same for independent, non-associated, and also uncorrelated random variables, it suffices to have a collection of k-wise uncorrelated random variables to obtain the following result.

Theorem 3.47 *If S is the sum of k-wise uncorrelated random variables, each of which is confined to the interval $[0, 1]$, then we have:*

(1) For $\delta \leq 1$,

   (a) if $k \leq \lfloor \delta^2\mu \cdot \mathrm{e}^{-1/3} \rfloor$, then $\Pr[|S - \mu| \geq \delta\mu] \leq \mathrm{e}^{-\lfloor k/2 \rfloor}$, and

   (b) if $k = \lfloor \delta^2\mu \cdot \mathrm{e}^{-1/3} \rfloor$, then $\Pr[|S - \mu| \geq \delta\mu] \leq \mathrm{e}^{-\lfloor \delta^2\mu/3 \rfloor}$.

(2) For $\delta \geq 1$,

(a) if $k \le \lfloor \delta\mu \cdot \mathrm{e}^{-1/3} \rfloor$*, then* $\Pr[|S-\mu| \ge \delta\mu] \le \mathrm{e}^{-\lfloor k/2 \rfloor}$*, and*

(b) if $k = \lfloor \delta\mu \cdot \mathrm{e}^{-1/3} \rfloor$*, then* $\Pr[|S-\mu| \ge \delta\mu] \le \mathrm{e}^{-\lfloor \delta\mu/3 \rfloor}$.

(3) If $\delta \ge 1$ *and* $k = \lceil \delta\mu \rceil$,

$$\Pr[|S-\mu| \ge \delta\mu] \le \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}} \right)^{\mu} .$$

In the case of k-wise negatively correlated random variables, the same bounds hold if $\Pr[|S-\mu| \ge \delta\mu]$ is replaced by $\Pr[S-\mu \ge \delta\mu]$. This is due to the fact that the expected values of the multilinear polynomials used in the proof are at most their expected values for the case that the random variables are k-wise uncorrelated.

For the case of k-wise self-weakening random variables with parameter λ (recall the definition in Section 3.2.6), all bounds for $\Pr[S-\mu \ge \delta\mu]$ given in Theorem 3.47 still hold, but with a factor of λ in front of each bound. The reason for this is that the expected values of the multilinear polynomials used in the proof have to be increased by a factor of λ, compared to the case of k-wise uncorrelated random variables.

Other types of dependent random variables

Suppose that T is a random variable that can be written in the form

$$T = p_1T_1 + p_2T_2 + \ldots + p_NT_N ,$$

where each T_i is a sum of independent random variables and $p_1, p_2, \ldots, p_N$ are non-negative numbers with $p_1 + p_2 + \ldots + p_N = 1$. The random variables T_i need not to be (mutually or even pairwise) independent. Then the following theorem holds

Theorem 3.48 ([Hoe63]) *If* T *is a random variable with properties defined above, then for every* $t \ge 0$ *and* $h > 0$,

$$\Pr[T \ge t] \le \sum_{i=1}^{N} p_i \mathrm{E}[\mathrm{e}^{hT_i}]/\mathrm{e}^{ht} .$$

$\mathrm{E}[\mathrm{e}^{hT_i}]$ may be obtained by using one of the bounds derived in the proofs of the previous theorems.

Proof of Theorem 3.48: For any $h > 0$,

$$\Pr[T \ge t] \le \mathrm{e}^{-ht}\mathrm{E}[\mathrm{e}^{hT}] . \tag{3.34}$$

Since the exponential function is convex, we have by Jensen's inequality

$$\exp(hT) = \exp\left(h \sum_{i=1}^{N} p_i T_i \right) \le \sum_{i=1}^{N} p_i \exp(hT_i) .$$

Using this in inequality (3.34) yields the theorem. □

Another technique to model dependent random variables is to use so-called dependency graphs. Consider any collection $X_1, \ldots, X_n$ of random variables. A directed graph $G = ([n], E)$ is called a *dependency graph* of $X_1, \ldots, X_n$, if edges are selected in this graph such that, for all $i \in [n]$, X_i is independent of any set of X_j's with $\{i,j\} \notin E$. Thus, any independent set of nodes in G represents a set of mutually independent random variables. In the following, we present techniques to find good tail estimates in such a situation.

Theorem 3.49 *Let $X_1, \ldots, X_n$ are binary random variables with a dependency graph of degree at most d. Then, for any $\delta > 0$,*

$$\Pr[S \geq (1+\delta)(d+1)\mu] \leq \left(\frac{\mathrm{e}^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} .$$

Proof. Let G be a dependency graph of $X_1, \ldots, X_n$ of degree at most d. Furthermore, let the random variable S' denote the size of the index set $I \subseteq [n]$ of maximum cardinality with the property that

- I is an independent set in G and
- all X_i with $i \in I$ are equal to 1.

According to the Loéve inequality, for any $\delta > 0$ and $h > 0$,

$$\Pr[S' \geq (1+\delta)\mu] \leq \mathrm{e}^{-h(1+\delta)\mu} \cdot \mathrm{E}[\mathrm{e}^{hS'}] .$$

Now, let $Y_1, \ldots, Y_n$ be independent binary random variables with $\mathrm{E}[Y_i] = \mathrm{E}[X_i]$ for all i and let $T = \sum_{i=1}^{n} Y_i$. Using the identity $\mathrm{e}^x = \sum_{i=0}^{\infty} x^i/i!$, it follows that $\mathrm{E}[\mathrm{e}^{hS'}] \leq \mathrm{E}[\mathrm{e}^{hT}]$, since all products of X_i's in $\mathrm{E}[\mathrm{e}^{hS'}]$ that are not independent have an expected value of 0, whereas in $\mathrm{E}[\mathrm{e}^{hT}]$ the corresponding products have an expected value of at least 0. Hence, using the same technique as in the proof of Theorem 3.35, we obtain for every $\delta > 0$

$$\Pr[S' \geq (1+\delta)\mu] \leq \mathrm{e}^{-h(1+\delta)\mu} \cdot \mathrm{E}[\mathrm{e}^{hT}] \leq \left(\frac{\mathrm{e}^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}$$

if h is set to $\ln(1+\delta)$.

For any $k \in \mathbb{N}_0$ it clearly holds that if $S' = k$ then S can be at most $k(d+1)$, since otherwise there must be at least $k+1$ independent X_i's that are 1. Thus,

$$\Pr[S \geq (1+\delta)(d+1)\mu] \leq \Pr[S' \geq (1+\delta)\mu] ,$$

which concludes the proof. □

If the X_i's are identically distributed, Rabani and Tardos gave a better tail estimate [RT96]. We improve their technique to non-identically distributed random variables.

Theorem 3.50 *If $X_1, \ldots, X_n$ are binary random variables with a dependency graph of degree at most d and $\mathrm{E}[X_i] \leq p$ for all i, then for any $\delta > 0$,*

$$\Pr[S \geq (1+2\delta)\mu + 4\delta p d] \leq 4d \cdot \mathrm{e}^{-\frac{\delta^2 \mu}{4d(1+\delta/3)}} .$$

Proof. We will use the following vital fact.

Fact 3.51 *For any sequence of non-negative reals $a_1, \ldots, a_n$ with $A = \sum_{i=1}^{n} a_i$ and $a_i \leq m$ for all $i \in [n]$ and any $k \in \mathbb{N}$, there are integers $1 \leq i_1 < i_2 < \ldots < i_{k-1} \leq n$ such that, with $i_0 = 1$ and $i_k = n+1$, $\sum_{j=i_\ell}^{i_{\ell+1}-1} a_j \leq A/k + m$ for all $0 \leq \ell \leq k-1$.*

According to Claim 2.1, any graph of degree d can be partitioned into at most $d+1$ independent sets. Let $I_1, \ldots, I_{d+1}$ be the corresponding sets in the dependency graph of $X_1, \ldots, X_n$. Fact 3.51 implies that for any $k \in \mathbb{N}$, $I_1, \ldots, I_{d+1}$ can be split into $m \le k \cdot d + (d+1)$ independent sets $I'_1, \ldots, I'_m$ with the property that

$$\mu_j = \mathrm{E}[\sum_{i \in I'_j} X_i] \le \frac{\mu}{k \cdot d} + p$$

for all $1 \le j \le m$. Let us choose $k = 2$, and let $S_j = \sum_{i \in I'_j} X_i$ for all j. Since I'_j is an independent set, its random variables are independent and therefore, according to the Chernoff bounds,

$$\begin{aligned} \Pr[S_j \ge \mu_j + \delta(\mu/(2d) + p)] &= \Pr[S_j \ge (1+\delta')\mu_j] \quad \text{with } \delta' = \delta(\mu/(2d)+p)/\mu_j \\ &\le \mathrm{e}^{-\frac{(\delta')^2 \mu_j}{2(1+\delta'/3)}} \\ &\le \mathrm{e}^{-\frac{\delta^2 \mu/(2d)}{2(1+\delta/3)}} . \end{aligned}$$

The last equality follows from the fact that $\frac{(\delta x/y)^2 y}{2(1+(\delta x/y)/3)} \ge \frac{\delta^2 x}{2(1+\delta/3)}$ for any $\delta, x, y > 0$ with $x \ge y$. Thus we obtain

$$\Pr[S \ge (1+2\delta)\mu + 4\delta p d] \le \Pr[\bigvee_{j \in [m]} (S_j \ge \mu_j + \delta(\mu/(2d)+p))] \le 4d \cdot \mathrm{e}^{-\frac{\delta^2 \mu/(2d)}{2(1+\delta/3)}} .$$

□

Dependency graphs are very useful as we will see later, but in the case that every random variable weakly depends on every other, Theorems 3.48 and 3.50 might result in far too bad bounds. In such a situation, the following theorem may still help.

Theorem 3.52 *For any k and any set of random variables $X_1, \ldots, X_n$ with $X_i \in [0,1]$ and the property that*

$$P_j(X_1, \ldots, X_n) \le P_j(p, \ldots, p)$$

for all $j \le k$, where P_j is defined as in the proof of Theorem 3.46, the same probability bounds hold for $\Pr[S_n \ge (1+\delta)\mu]$ as given in Theorem 3.46 for the case that $X_1, \ldots, X_n$ are k-wise independent random variables with $\mathrm{E}[S_n] = \mu = p \cdot n$.

Despite of its innocent-looking nature, this theorem is extremely powerful. All that has to be shown in order for $P_j(X_1, \ldots, X_n) \le P_j(p, \ldots, p)$ to hold for all $j \le k$ is that for all $j \le k$ we have

$$\mathrm{E}[\textstyle\prod_{i \in S} X_i] \le p^j$$

for a set S chosen uniformly at random out of the set of all subsets of $[n]$ of size j. In order to demonstrate that Theorem 3.52 is useful, we give the following example.

Example: Consider the problem of embedding a graph $G = (V, E)$ of maximum degree d into a graph $H = (W, F)$ with $|V| = |W| = n$. In particular, we are searching for a way to map the nodes of G one-to-one to the nodes of H and to select paths in H simulating the edges of G such that the congestion and dilation of the path collection is as small as possible.

Let R be the routing number of H (for the definition of the routing number, see Example 2 in Section 3.2.5). To facilitate the selection of the paths, we assume that we are given a system of paths $P = (p_{u,v})_{u,v \in W}$ with dilation at most R and congestion at most $(n-1)R$. That such a system of paths indeed exists can be shown quite easily: take any best possible path collection for the permutation routing problem defined by π_i with $\pi_i(x) = (x+i) \bmod n$ for all $i \in \{1, \ldots, n-1\}$ and $x \in \{0, \ldots, n-1\}$, and define the union of these path collections as the system of paths we intend to use. To find a mapping of G to H, we perform the following random experiment:

Choose a permutation π uniformly at random from the set of all possible permutations on $[n]$. Given π, every node $v \in V$ is mapped to node $\pi(v)$ in W, and every edge $(v, w) \in E$ is represented by the path $p_{\pi(v),\pi(w)}$ in H.

Clearly, the dilation of the paths used for the embedding is at most R. Hence, it remains to bound the congestion of the selected paths. Let e be any fixed edge in H. We start with bounding the expected congestion.

For every pair of nodes $u, v \in W$ with $u \neq v$, let the binary random variable $X_{u,v}$ be 1 if and only if e is contained in $p_{u,v}$ and $(\pi^{-1}(u), \pi^{-1}(v)) \in E$. Furthermore, let $X = \sum_{u,v \in W} X_{u,v}$. Thus X denotes the congestion caused at e by the selected paths. Since π is chosen uniformly at random and the degree of G is d, the probability that for some given path $p_{u,v}$ we have $(\pi^{-1}(u), \pi^{-1}(v)) \in E$ is at most

$$\frac{d \cdot n}{n(n-1)} = \frac{d}{n-1} \; .$$

Since at most $(n-1)R$ paths are crossing e,

$$\mathrm{E}[X] \leq (n-1)R \cdot \frac{d}{n-1} = d \cdot R \; .$$

Next we show that w.h.p. the congestion at e is not far away from $\mathrm{E}[X]$. This is trivial for $d > n/2$. Hence, in the following we assume that $d \leq n/2$. Since there may be high dependencies among the $X_{u,v}$, this cannot be done by simply using Chernoff bounds. We first need the following claim.

Claim 3.53 *Consider any fixed $(u, v) \in W^2$ and any set of pairs $(u_1, v_2), \ldots, (u_k, v_k) \in W^2$ with $k \geq 1$. If it holds that either u or v is a node that has not been used in any pair (u_i, v_i), then*

$$\Pr[X_{u,v} = 1 \mid X_{u_1,v_1} = \ldots = X_{u_k,v_k} = 1] \leq \frac{d}{n-2k-1} \; .$$

Proof. Let m be the number of distinct nodes in the pairs (u_i, v_i), $i \in [k]$. First we consider the case that u is (and therefore v is not) identical to some node u_i or v_i, $i \in [k]$ (or vice versa). In this case, there are $n - m$ nodes of G that can be embedded in v, and at most d of them are edges in G. Since a mapping of G to H is chosen uniformly at random, we therefore have

$$\Pr[X_{u,v} = 1 \mid X_{u_1,v_1} = \ldots = X_{u_k,v_k} = 1] \leq \frac{d}{n-m} \leq \frac{d}{n-2k} \; .$$

For the case that u and v are not equal to any node u_i or v_i, $i \in [k]$, the number of pairs of nodes in G that can be embedded in (u, v) is equal to $(n-m)(n-m-1)$. Moreover, the number of remaining edges in G that can be represented by $p_{u,v}$ is at most $d(n-m)$. Thus we obtain

$$\begin{aligned} \Pr[X_{u,v} = 1 \mid X_{u_1,v_1} = \ldots = X_{u_k,v_k} = 1] &\leq \frac{d(n-m)}{(n-m)(n-m-1)} \\ &= \frac{d}{n-m-1} \leq \frac{d}{n-2k-1} \; , \end{aligned}$$

which concludes the proof. □

Now, consider the random experiment of choosing a set $\{(u_1, v_1), \ldots, (u_k, v_k)\}$ of pairs of distinct nodes uniformly at random among all sets of pairs of distinct nodes of size k. Since the probability that both u_i and v_i are already used in the pairs $(u_1, v_1), \ldots, (u_{i-1}, v_{i-1})$ is at most

$$\frac{(2(i-1))(2(i-1)-1)-(i-1)}{n(n-1)-(i-1)} \leq \frac{4k^2}{n^2}$$

if k is sufficiently small compared to n, it follows from Claim 3.53 that

$$\Pr[X_{u_i,v_i} = 1 \mid X_{u_1,v_1} = \ldots = X_{u_{i-1},v_{i-1}} = 1] \leq \frac{4k^2}{n^2} + \frac{d}{n-2k-1} ,$$

where the probability is taken according to the random choice of (u_i, v_i) and the random mapping of G to H. Thus,

$$\Pr[X_{u_1,v_1} = \ldots = X_{u_k,v_k} = 1] \leq \left(\frac{4k^2}{n^2} + \frac{d}{n-2k-1}\right)^k$$

In the notation of Theorem 3.52, this implies that

$$P_k(\{X_{u,v} \;:\; u \neq v\}) \leq P_k(p_k, \ldots, p_k) \quad \text{where} \quad p_k = \frac{4k^2}{n^2} + \frac{d}{n-2k-1} .$$

For $k = (\sqrt{n}-1)/2$ we have $p_j \leq (d+1)/(n-\sqrt{n})$ for all $j \leq k$. Hence,

$$P_j(\{X_{u,v} \;:\; u \neq v\}) \leq P_j(p, \ldots, p) \quad \text{with} \quad p = \frac{d+1}{n-\sqrt{n}} \tag{3.35}$$

for all $j \leq k$. Let $\mu = p \cdot (n-1)R$. Using (3.35) in Theorem 3.52 yields that, for any $\delta > 0$,

$$\Pr[S \geq (1+\delta)\mu] \leq \mathrm{e}^{-\min[\delta^2\mu/3,\ \delta\mu/3,\ k/2]} .$$

Hence, the probability can be made as low as $1/n^3$ that the congestion at edge e exceeds $(d+1)R + O(\max[\sqrt{(d+1)R\log n},\ \log n])$. Since H can have at most n^2 edges, the probability that the congestion of the embedding exceeds the same bound can be made less than $1/n$.

We note that Valiant's trick (see Example 2 in Section 3.2.5) would have enabled us to use a much simpler proof to obtain a congestion of $O(d \cdot R + \log n)$, w.h.p, for the embedding. However, using this trick, it is at best possible to show a bound of $2d \cdot R$, whereas the proof above shows that a bound of approximately $(d+1)R$ can be reached.

3.3.4 Proofs of the Theorems in Section 3.3

Proof of Theorem 3.35: It remains to prove inequality (3.21). For $0 \leq \delta \leq 1$ and $h > 0$,

$$\begin{aligned}
\Pr[S_n \leq (1-\delta)\mu] \quad &= \quad \Pr[-S_n \geq -(1-\delta)\mu] = \Pr[\mathrm{e}^{-hS_n} \geq \mathrm{e}^{-h(1-\delta)\mu}] \\
&\overset{\text{Markov}}{\leq} \quad \mathrm{e}^{h(1-\delta)\mu} \cdot \mathrm{E}[\mathrm{e}^{-h \cdot S_n}]
\end{aligned} \tag{3.36}$$

and

$$\begin{aligned}
\mathrm{E}[\mathrm{e}^{-h\cdot S_n}] &= \mathrm{E}[\mathrm{e}^{-h(X_1+\ldots+X_n)}] = \mathrm{E}[\mathrm{e}^{-h\cdot X_1}\cdots \mathrm{e}^{-h\cdot X_n}] = \prod_{i=1}^n \mathrm{E}[\mathrm{e}^{-h\cdot X_i}] \\
&= \prod_{i=1}^n (p_i\mathrm{e}^{-h} + (1-p_i)) = \prod_{i=1}^n (1-p_i(1-\mathrm{e}^{-h})) \le \prod_{i=1}^n \mathrm{e}^{-p_i(1-\mathrm{e}^{-h})} \\
&= \mathrm{e}^{-\mu(1-\mathrm{e}^{-h})} .
\end{aligned}$$

This yields together with inequality (3.36) that

$$\Pr[S_n \le (1-\delta)\mu] \le \mathrm{e}^{h(1-\delta)\mu}\mathrm{e}^{-\mu(1-\mathrm{e}^{-h})} = \mathrm{e}^{-(1-h(1-\delta)-\mathrm{e}^{-h})\mu} . \tag{3.37}$$

The right-hand side of (3.37) attains its minimum at $h = h_1$, where $h_1 = -\ln(1-\delta)$. Inserting this in (3.37) yields

$$\Pr[S_n \le (1-\delta)\mu] \le (1-\delta)^{-(1-\delta)\mu} \cdot \mathrm{e}^{-\delta\cdot\mu} = \left(\frac{\mathrm{e}^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu} ,$$

which completes the proof of Theorem 3.35. □

Proof of Theorem 3.36: We will only prove inequality (3.25). The proofs of the other inequalities can be found in [Hoe63].

Let X be a random variable such that $a \le X \le b$. Since the exponential function $\exp(hX)$ is convex, its graph is bounded above on the interval $[a, b]$ by the straight line which connects its ordinates at $X = a$ and $X = b$. Thus,

$$\mathrm{e}^{hX} \le \frac{b-X}{b-a}\cdot \mathrm{e}^{h\cdot a} + \frac{X-a}{b-a}\cdot \mathrm{e}^{h\cdot b} .$$

Hence, we obtain

$$\mathrm{E}[\mathrm{e}^{hX}] \le \frac{b-\mathrm{E}[X]}{b-a}\cdot \mathrm{e}^{h\cdot a} + \frac{\mathrm{E}[X]-a}{b-a}\cdot \mathrm{e}^{h\cdot b} . \tag{3.38}$$

Similar to the proof of Theorem 3.35 it holds for any $h > 0$ that

$$\Pr[A_n \ge t] \le \mathrm{e}^{-h\cdot n\cdot t} \cdot \prod_{i=1}^n \mathrm{E}[\mathrm{e}^{hX_i}] . \tag{3.39}$$

By assumption, $0 \le X_i \le 1$. Let $\mu_i = \mathrm{E}[X_i]$. Using inequality (3.38), we have

$$\prod_{i=1}^n \mathrm{E}[\mathrm{e}^{hX_i}] \le \prod_{i=1}^n (1-\mu_i+\mu_i\mathrm{e}^h) .$$

Since the geometric mean does not exceed the arithmetic mean (see Fact 3.4),

$$\left[\prod_{i=1}^n (1-\mu_i+\mu_i\mathrm{e}^h)\right]^{1/n} \le \frac{1}{n}\sum_{i=1}^n (1-\mu_i+\mu_i\mathrm{e}^h) = 1-\bar{\mu}+\bar{\mu}\mathrm{e}^h . \tag{3.40}$$

It follows from (3.39) that

$$\Pr[A_n \ge t] \le \left[\mathrm{e}^{-h\cdot t}(1-\bar{\mu}+\bar{\mu}\mathrm{e}^h)\right]^n . \tag{3.41}$$

The right hand side of (3.41) attains its minimum at $h = h_0$, where

$$h_0 = \ln \frac{(1-\bar{\mu})t}{(1-t)\bar{\mu}} .$$

Since $\bar{\mu} < t < 1$, h_0 is positive. Inserting h_0 in (3.41) yields inequality (3.25). $\square$

Proof of Theorem 3.38: We start with proving inequality (3.28). According to the Loéve inequality it holds that

$$\Pr[S_n \geq (1+\delta)\mu] \leq \mathrm{e}^{-h(1+\delta)\mu} \cdot \mathrm{E}[\mathrm{e}^{h \cdot S}] . \tag{3.42}$$

Since $X_1, \ldots, X_n$ are independent, it follows that

$$\begin{aligned}
\mathrm{E}[\mathrm{e}^{h\cdot S}] &= \mathrm{E}[\mathrm{e}^{h\cdot X_1} \cdots \mathrm{e}^{h \cdot X_n}] = \prod_{i=1}^{n} \mathrm{E}[\mathrm{e}^{h\cdot X_i}] = \prod_{i=1}^{n} \left(\sum_{j \geq 1} (1-p_i)^{j-1} p_i \cdot \mathrm{e}^{h \cdot j} \right) \\
&= \prod_{i=1}^{n} \left(p_i \cdot \mathrm{e}^h \sum_{j \geq 0} ((1-p_i)\cdot \mathrm{e}^h)^j \right) .
\end{aligned}$$

In order for the right-hand side to be finite we require that $\mathrm{e}^h < 1/(1-p_i)$ for all $i \in [n]$, or equivalently, $\mathrm{e}^h < 1/(1-p)$. In this case, we obtain for $\mu_i = \mathrm{E}[X_i]$ for all $i \in [n]$

$$\mathrm{E}[\mathrm{e}^{h \cdot S}] = \prod_{i=1}^{n} \left(p_i \mathrm{e}^h \cdot \frac{1}{1-(1-p_i)\mathrm{e}^h} \right) = \prod_{i=1}^{n} \frac{1}{1-\mu_i + \mu_i \mathrm{e}^{-h}} .$$

Thus together with (3.42) we have

$$\Pr[S \geq (1+\delta)\mu] \leq \prod_{i=1}^{n} \left(\frac{\mathrm{e}^{-h(1+\delta)\mu_i}}{1-\mu_i+\mu_i \mathrm{e}^{-h}} \right) .$$

For $\mu_i = 1/p$, the fraction $\mathrm{e}^{-h(1+\delta)\mu_i}/(1-\mu_i+\mu_i\mathrm{e}^{-h})$ attains its minimum for $h = h_0 = \ln(1-\delta p/(1+\delta))^{-1}$, which fulfills the condition that $\mathrm{e}^h < 1/(1-p)$ for all $\delta > 0$. For $h = h_0$ we have

$$\begin{aligned}
\frac{\mathrm{e}^{-h(1+\delta)\mu_i}}{1-\mu_i+\mu_i\mathrm{e}^{-h}} &= \left(1 - \frac{\delta p}{1+\delta}\right)^{(1+\delta)\mu_i} \cdot \frac{1}{1-\mu_i \cdot \frac{\delta p}{1+\delta}} \\
&\leq \mathrm{e}^{-\delta p \mu_i} \cdot \left(1 + \frac{\delta p \mu_i}{1+\delta(1-p\mu_i)}\right) \leq (1+\delta p \mu_i) \cdot \mathrm{e}^{-\delta p \mu_i} .
\end{aligned}$$

For any vector $\boldsymbol{x} = (x_1, \ldots, x_2) \in \mathrm{IR}_+^n$, let the function $f : \mathrm{IR}_+^n \to \mathrm{IR}_+$ be defined as $f(\boldsymbol{x}) = \prod_{i=1}^n (1+x_i)$. It is not difficult to show that f is Schur-concave. Hence,

$$\begin{aligned}
\Pr[S \geq (1+\delta)\mu] &\leq \prod_{i=1}^{n} \left((1+\delta p \mu_i) \mathrm{e}^{-\delta p \mu_i} \right) \\
&\leq \left(\frac{\sum_i (1+\delta p \mu_i)}{n} \right)^n \mathrm{e}^{-\sum_i \delta p \mu_i} = \left(1 + \frac{\delta \mu p}{n}\right)^n \cdot \mathrm{e}^{-\delta \mu p} .
\end{aligned}$$

Next we prove inequality (3.29). Using the Markov inequality (see the proof of Theorem 3.35), it holds that

$$\Pr[S \leq (1-\delta)\mu] \leq \mathrm{e}^{h(1-\delta)\mu} \cdot \mathrm{E}[\mathrm{e}^{-h \cdot S}] . \tag{3.43}$$

Since $X_1, \ldots, X_n$ are independent, it follows that

$$\begin{aligned}
\mathrm{E}[\mathrm{e}^{-h\cdot S}] &= \prod_{i=1}^{n} \mathrm{E}[\mathrm{e}^{-h\cdot X_i}] = \prod_{i=1}^{n} \left[\sum_{j\ge 1}(1-p_i)^{j-1} p_i \mathrm{e}^{-hj}\right] \\
&= \prod_{i=1}^{n} \left[\frac{p_i}{\mathrm{e}^h} \sum_{j\ge 0} \left(\frac{1-p_i}{\mathrm{e}^h}\right)^j\right] = \prod_{i=1}^{n} \frac{p_i}{\mathrm{e}^h} \cdot \frac{1}{1-(1-p_i)/\mathrm{e}^h}
\end{aligned}$$

for all $h \ge 0$. In this case,

$$\mathrm{E}[\mathrm{e}^{-h\cdot S}] = \prod_{i=1}^{n} \frac{1}{\mathrm{e}^h/p_i + 1 - 1/p_i} = \prod_{i=1}^{n} \frac{1}{1-\mu_i+\mu_i \mathrm{e}^h} . \tag{3.44}$$

Inserting this in (3.43) yields

$$\Pr[S \le (1-\delta)\mu] \le \prod_{i=1}^{n} \left(\frac{\mathrm{e}^{h(1-\delta)\mu_i}}{1-\mu_i+\mu_i \mathrm{e}^h}\right) \tag{3.45}$$

The fraction on the right-hand side of (3.45) is minimal for $h = h_1$, where

$$h_1 = \ln\left(1 + \frac{\delta}{\mu_i(1-\delta)}\right) .$$

We choose $h = \ln(1 + \delta p/(1-\delta))$. For this we have

$$\Pr[S \le (1-\delta)\mu] \le \prod_{i=1}^{n} \left(\frac{(1+\delta p/(1-\delta))^{(1-\delta)\mu_i}}{1+\delta p \mu_i/(1-\delta)}\right) .$$

Since $p\mu_i \le 1$ it holds that

$$\left(1 + \frac{\delta p}{1-\delta}\right)^{(1-\delta)\mu_i} \le \mathrm{e}^{\delta p \mu_i} \le 1 + \delta p \mu_i \sum_{i\ge 0} \tfrac{\delta^i}{(i+1)!} .$$

Thus,

$$\begin{aligned}
\frac{(1+\delta p/(1-\delta))^{(1-\delta)\mu_i}}{1+\delta p\mu_i/(1-\delta)} &\le \frac{1+\delta p\mu_i \sum_{i\ge 0}\frac{\delta^i}{(i+1)!}}{1+\delta p\mu_i\sum_{i\ge 0}\delta^i} \\
&\le 1-(1-\delta)\delta p\mu_i\Big[\Big(\textstyle\sum_{i\ge 0}\delta^i\Big) - \Big(\textstyle\sum_{i\ge 0}\frac{\delta^i}{(i+1)!}\Big)\Big] ,
\end{aligned}$$

since

$$\begin{aligned}
&1+\delta p\mu_i \sum_{i\ge 0}\frac{\delta^i}{(i+1)!} \\
&\le 1+\delta p\mu_i\sum_{i\ge 0}\delta^i - (1-\delta)\left[\left(1+\delta p\mu_i\sum_{i\ge 0}\delta^i\right) - \left(1+\delta p\mu_i\sum_{i\ge 0}\frac{\delta^i}{(i+1)!}\right)\right]\left(1+\delta\sum_{i\ge 0}\delta^i\right) \\
&\le 1+\delta p\mu_i\sum_{i\ge 0}\delta^i - (1-\delta)\left[\delta p\mu_i\sum_{i\ge 0}\delta^i - \delta p\mu_i\sum_{i\ge 0}\frac{\delta^i}{(i+1)!}\right]\left(1+\delta p\mu_i\sum_{i\ge 0}\delta^i\right) \\
&= \left(1-(1-\delta)\delta p\mu_i\left[\sum_{i\ge 0}\delta^i - \sum_{i\ge 0}\frac{\delta^i}{(i+1)!}\right]\right)\left(1+\delta p\mu_i\sum_{i\ge 0}\delta^i\right) .
\end{aligned}$$

Hence,

$$\begin{aligned}\frac{(1+\delta p/(1-\delta))^{(1-\delta)\mu_i}}{1+\delta p\mu_i/(1-\delta)} &\leq \mathrm{e}^{-(1-\delta)\delta p\mu_i\left[\left(\sum_{i\geq 0}\delta^i\right)-\left(\sum_{i\geq 0}\frac{\delta^i}{(i+1)!}\right)\right]} \\ &\leq \mathrm{e}^{-(1-\delta)\delta p\mu_i\cdot\frac{1}{2}\sum_{i\geq 1}\delta^i} = \mathrm{e}^{-\delta^2 p\cdot\mu_i/2}\ .\end{aligned}$$

Inserting this in (3.45) yields

$$\Pr[S \leq (1-\delta)\mu] \leq \mathrm{e}^{-\delta^2 p\mu/2}\ .$$

□

Proof of Theorem 3.43: We only show inequality (3.31). For this we need two lemmas.

Lemma 3.54 *If X is a random variable such that $\mathrm{E}[X]=0$, $\mathrm{V}[X]=\nu$ and $X\leq b$, then for any positive number h*

$$\mathrm{E}[\mathrm{e}^{hX}] \leq \frac{b^2}{b^2+\nu}\cdot\mathrm{e}^{-(\nu/b)h}+\frac{\nu}{b^2+\nu}\cdot\mathrm{e}^{bh}\ .$$

A proof of this lemma can be found in Bennett [Ben62].

Lemma 3.55 *If $c>0$, the function*

$$f(x)=\ln\left(\frac{1}{1+x}\mathrm{e}^{-cx}+\frac{x}{1+x}\mathrm{e}^{c}\right)$$

has a negative second derivative for $x\geq 0$.

For a proof of this lemma see [Hoe63]. By assumption, $\mathrm{E}[X_i]=0$ and $X_i\leq b$. Let $\nu_i=\mathrm{V}[X_i]$. From Lemma 3.54 it follows that

$$\begin{aligned}\Pr[A_n\geq t] &\leq \mathrm{e}^{-h\cdot n\cdot t}\prod_{i=1}^{n}\left(\frac{b^2}{b^2+\nu_i}\cdot\mathrm{e}^{-(\nu_i/b)h}+\frac{\nu_i}{b^2+\nu_i}\cdot\mathrm{e}^{bh}\right) \\ &= \mathrm{e}^{-h\cdot n\cdot t+\sum_{i=1}^{n}f(\nu_i/b^2)}\ ,\end{aligned}$$

where f is the function defined in Lemma 3.55 with $c=bh$. Since, by Lemma 3.55, $f(x)$ has a negative second derivative, $-f(x)$ is convex for $x\geq 0$. Therefore by Jensen's inequality

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}f(\nu_i/b^2) &\leq f\left(\frac{1}{n}\sum_{i=1}^{n}\frac{\nu_i}{b^2}\right)=f(\bar\nu/b^2) \\ &= \ln\left(\frac{b^2}{b^2+\bar\nu}\mathrm{e}^{-(\bar\nu/b)h}+\frac{\bar\nu}{b^2+\bar\nu}\mathrm{e}^{bh}\right)\ .\end{aligned}$$

It follows that

$$\Pr[A_n\geq t]\leq\left(\frac{b^2}{b^2+\bar\nu}\mathrm{e}^{-(t+\bar\nu/b)h}+\frac{\bar\nu}{b^2+\bar\nu}\mathrm{e}^{(b-t)h}\right)^n\ . \tag{3.46}$$

The right-hand side of this inequality attains it minimum at $h=h_0$, where

$$h_0=\frac{b}{b^2+\bar\nu}\ln\frac{1+tb/\bar\nu}{1-t/b}\ .$$

Inserting this in (3.46), we obtain inequality (3.31) of Theorem 3.43. □

Proof of Theorem 3.46: We will only prove the case when $\delta \geq 1$ and $k = \lceil \delta\mu \rceil$. First we assume that we have n binary random variables $X_1, \ldots, X_n$. Instead of using functions of the form e^{hS} to achieve good probability bounds, we intend to use a family of symmetric multilinear polynomials $P_j(\boldsymbol{z})$ with $j \in [n]$ and $\boldsymbol{z} = (z_1, \ldots, z_n) \in \mathrm{IR}^n$, where

$$P_j(\boldsymbol{z}) = \sum_{i_1 < i_2 < \ldots < i_j} z_{i_1} z_{i_2} \cdots z_{i_j} \; .$$

If $X_1, \ldots, X_n$ are independent, then $\mathrm{E}[P_j(X_1, \ldots, X_n)]$ is explicitly available:

$$\mathrm{E}[P_j(X_1, \ldots, X_n)] = P_j(p_1, \ldots, p_n) \; ,$$

where $p_i = \Pr[X_i = 1]$.

First, we will demonstrate that using multilinear polynomials is a very general approach that includes the exponential moment method. Clearly, for any sequence of variables $z_1, \ldots, z_n$ in $\{0, 1\}$ and any positive integer j,

$$(z_1 + z_2 + \ldots + z_n)^j = \sum_{i=1}^{\min[j,n]} a_i P_i(z_1, \ldots, z_n) \tag{3.47}$$

for some non-negative reals $a_1, \ldots, a_{\min[j,n]}$ with $\sum_i a_i \binom{n}{i} = n^j$. Let $S = \sum_{i=1}^n X_i$. Due to the fact that the X_i's only take values in $\{0, 1\}$ and the a_i's are limited by n^j, the expansion $e^{hS} = \sum_{j=1}^{\infty} (h^j/j!) \cdot S^j$ is finite. Since all of the coefficients $(h^j/j!)$ are positive if $h > 0$, we obtain the following lemma.

Lemma 3.56 *Let $z_1, \ldots, z_n \in \{0, 1\}$. Then, for any $h > 0$, there are non-negative reals $a_1, \ldots, a_n$ such that*

$$\mathrm{e}^{h(z_1 + z_2 + \ldots + z_n)} = \sum_{i=0}^{n} a_i P_i(z_1, \ldots, z_n) \; .$$

Hence, the family of functions $\{\mathrm{e}^{hX} \; : \; h > 0\}$ is just a special case of the family of polynomials we use here.

Now we show how to obtain tail estimates using multilinear polynomials. For any $\boldsymbol{y} = (y_0, y_1, \ldots, y_n) \in \mathrm{IR}_+^{n+1}$ and $\boldsymbol{z} = (z_1, \ldots, z_n) \in \mathrm{IR}^n$, define $f_{\boldsymbol{y}}(\boldsymbol{z}) = \sum_{i=0}^n y_i P_i(\boldsymbol{z})$. With this notation, Lemma 3.56 can be restated as $\forall h \; \exists \boldsymbol{y}$: $f_{\boldsymbol{y}}(X_1, \ldots, X_n) = \mathrm{e}^{h \cdot S}$. Clearly, for any non-negative integer m, $S = m$ if and only if

$$f_{\boldsymbol{y}}(X_1, \ldots, X_n) = \sum_{i=0}^{m} y_i \binom{m}{i} \tag{3.48}$$

and hence,

$$\Pr[S \geq t] \;\; = \;\; \Pr\left[f_{\boldsymbol{y}}(X_1, \ldots, X_n) \geq \sum_{i=0}^{t} y_i \binom{t}{i} \right] \leq \frac{\mathrm{E}[f_{\boldsymbol{y}}(X_1, \ldots, X_n)]}{\sum_{i=0}^{t} y_i \binom{t}{i}} \; . \tag{3.49}$$

Our goal is to find a $\boldsymbol{y}$ for which the right-hand side of this formula is minimal. We will restrict our attention to those $\boldsymbol{y}$ with $y_{t+1} = \ldots = y_n = 0$. To find a best possible assignment for the rest, we state the following lemma. For a proof see [SSS95]. (It essentially has to be shown that $P_i(.)$ is Schur-concave.)

Lemma 3.57 *For any $i > 0$ and $s \geq 0$, $P_i(z_1, \ldots, z_n)$ is maximized by setting $z_1 = \ldots = z_n = s/n$, when subject to the constraints that $z_1, \ldots, z_n \in \mathsf{IR}_+$ and $\sum_{i=1}^n z_i = s$.*

Inequality (3.49) and Lemma 3.57 imply that if $p = (\sum_{i=1}^n p_i)/n = \mu/n$, then for any $y \in \mathsf{IR}_+^{n+1}$,

$$\Pr[X \geq t] \leq \frac{\sum_{i=0}^t y_i \binom{n}{i} p^i}{\sum_{i=0}^t y_i \binom{t}{i}} \ .$$

This is minimized by setting $y_i = 1$ if $i = i^*$ and 0 otherwise, where i^* is the integer at which $\binom{n}{i} p^i / \binom{t}{i}$ is minimized. Since

$$\frac{\binom{n}{i+1} p^{i+1} / \binom{t}{i+1}}{\binom{n}{i} p^i / \binom{t}{i}} = \frac{(n-i)p}{t-i} \ ,$$

which is less than, equal to, or greater than 1 depending on if i is less than, equal to, or greater than $(t - np)/(1 - \mu/n)$, $\binom{n}{i} p^i / \binom{t}{i}$ is minimized at

$$i^* = \left\lceil \frac{t - np}{1 - \mu/n} \right\rceil = \left\lceil \frac{\delta\mu}{1 - \mu/n} \right\rceil$$

for $t = (1+\delta)\mu$. Since $i^* > \lceil \delta\mu \rceil$ and $X_1, \ldots, X_n$ are only $\lceil \delta\mu \rceil$-wise independent, we choose $i^* = \lceil \delta\mu \rceil$. In this case,

$$\begin{aligned}
\Pr[S \geq t] &\leq \binom{n}{i^*} p^{i^*} \Big/ \binom{t}{i^*} \\
&\leq \left(\frac{n}{t}\right)^{i^*} \left(\frac{\mu}{n}\right)^{i^*} \left(\frac{n}{n - i^*}\right)^{n - i^*} \left(\frac{t - i^*}{t}\right)^{t - i^*} \\
&\leq \left(\frac{n}{(1+\delta)\mu}\right)^{\delta\mu} \left(\frac{\mu}{n}\right)^{\delta\mu} \left(\frac{n}{n - \delta\mu}\right)^{n - \delta\mu} \left(\frac{\mu}{(1+\delta)\mu}\right)^{\mu} \\
&= \frac{(n/n - \delta\mu)^{n - \delta\mu}}{(1+\delta)^{(1+\delta)\mu}} = \left(\frac{1}{1+\delta}\right)^{(1+\delta)\mu} \left(1 + \frac{\delta\mu}{n - \delta\mu}\right)^{n - \delta\mu} \\
&\leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{\mu} \ .
\end{aligned}$$

So far, we have assumed that the X_i are binary random variables. For the case that X_i can take any real value in $[0,1]$, we need the following lemma.

Lemma 3.58 *Let z_i be real numbers with $0 \leq z_i \leq 1$ for all $i \in [n]$ and suppose that $m \geq 0$, $j \leq m$ and $\sum_{i=1}^n z_i \geq m$. Then,*

$$P_j(z_1, \ldots, z_n) \geq \binom{m}{j} \ .$$

Using this lemma in (3.48) yields the theorem. □

3.4 The Method of Bounded Differences

In this section, we present methods of finding tail estimates for more general functions than just sums of random variables.

3.4.1 Independent Bounded Differences Inequalities

First, we present tail estimates that are based on bounds on the difference of functions over independent random variables. The following result is due to McDiarmid [McD89].

Theorem 3.59 ([McD89]) *Let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be a vector of independent random variables with X_i taking values in a set A_i for each i. Suppose that the real-valued function f defined on $A_1 \times \ldots \times A_n$ satisfies*

$$|f(\boldsymbol{x}) - f(\boldsymbol{x}')| \leq c_i$$

whenever the vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$ differ only in the ith coordinate. Let μ be the expected value of $f(\boldsymbol{X})$. Then for any $d \geq 0$,

$$\Pr[f(\boldsymbol{X}) - \mu \geq d] \leq \mathrm{e}^{-2d^2/\sum_{i=1}^n c_i^2}$$

and

$$\Pr[f(\boldsymbol{X}) - \mu \leq -d] \leq \mathrm{e}^{-2d^2/\sum_{i=1}^n c_i^2} .$$

If we assume the X_i's to be binary random variables and $f(\boldsymbol{X}) = \sum_{i=1}^n X_i$, then we obtain

$$\Pr[f(\boldsymbol{X}) - \mu \geq d] \leq \mathrm{e}^{-2d^2/n} . \tag{3.50}$$

The problem with this result is that it only provides reasonable probability bounds for $d \geq \sqrt{n}$. To overcome this problem, we introduce the following notation.

Given a random variable X, its *median* $\mathrm{M}[X]$ is defined as a number with the property that

$$\Pr[X \leq \mathrm{M}[X]] \geq \frac{1}{2} \quad \text{and} \quad \Pr[X \geq \mathrm{M}[X]] \geq \frac{1}{2} .$$

Using the median of a random variable, Talagrand proved the following theorem (see [Tal95a, Mol98]).

Theorem 3.60 ([Tal95a]) *Let X be a random variable determined by n independent trials $T_1, \ldots, T_n$ and satisfying*

(1) changing the outcome of any one trial can affect X by at most c, and

(2) for any s, if $X \geq s$ then there are s trials $T_{i_1}, \ldots, T_{i_s}$ whose outcomes certify that $X \geq s$,

then for any $0 < d \leq \mathrm{M}[X]$,

$$\Pr[|X - \mathrm{M}[X]| > d] \leq 2\mathrm{e}^{-\frac{d^2}{16c^2\mathrm{M}[X]}} .$$

The fact that Talagrand's inequality proves concentration around the median rather than the expected value is not a serious problem, as in the situation where Talagrand's inequality applies, those two values are very close together, and so concentration around one implies concentration around the other:

Fact 3.61 *Under the conditions of Talagrand's inequality,*

$$|\mathrm{E}[X] - \mathrm{M}[X]| \leq 3c\sqrt{\mathrm{E}[X]} .$$

Using this fact and Talagrand's inequality in Theorem 3.59, we obtain the following result (see also [Mol98]).

Theorem 3.62 *Let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be a vector of independent random variables with X_i taking values in a set A_i for each i. Suppose that the real-valued function f defined on $A_1 \times \ldots \times A_n$ satisfies*

(1) changing the outcome of any one trial can affect f by at most c, and

(2) for any s, if $f(\boldsymbol{X}) \geq s$ then there are s variables $X_{i_1}, \ldots, X_{i_s}$ whose outcomes certify this,

then for any $0 < d \leq \mu$ with $\mu = \mathrm{E}[f(\boldsymbol{X})]$,

$$\Pr[|f(\boldsymbol{X}) - \mu| > d + 3c\sqrt{\mu}] \leq 2\mathrm{e}^{-\frac{d^2}{16c^2\mu}} .$$

Example: We will give a simple example of how to use McDiarmid's bounded differences method.

Claim 3.63 *Let n balls be thrown randomly into n bins. With high probability, at most $n/2^c$ balls are in bins with load at least c, for $8 \leq c \leq \log n/4$.*

Proof. We define the binary random variables X_i, $1 \leq i \leq n$, to be 1 if and only if ball i resides in a bin with a load of at least c. Obviously, it holds

$$\begin{aligned} \Pr[X_i = 1] &\leq \binom{n-1}{c-1}\left(\frac{1}{n}\right)^{c-1} \\ &\leq \left(\frac{\mathrm{e}}{c-1}\right)^{c-1} \leq \left(\frac{1}{2}\right)^{c+1} \qquad \text{for } c \geq 8. \end{aligned}$$

Thus, for $X = \sum_{i=1}^n X_i$ we obtain $\mathrm{E}[X] \leq n/2^{c+1}$. Now we use Theorem 3.59 to bound the deviation from the expected value. Let the random variable Y_j, $1 \leq j \leq n$, denote the random destination chosen by ball j. Then we can view X, the number of balls that reside in a bin of load at least c, as a function f in the Y_j's, i.e.,

$$X = f(\{Y_j,\ 1 \leq j \leq n\}) .$$

If the value of Y_j changes for any j, then X changes by at most c. Therefore, by Theorem 3.59, we obtain for $c \leq \log n/4$,

$$\Pr[X \geq \mathrm{E}[X] + \tfrac{n}{2^{c+1}}] \leq \mathrm{e}^{-\frac{2(n/2^c)^2}{nc^2}} \leq \mathrm{e}^{-\frac{2n^2}{\sqrt{n}\cdot n\cdot(\log n/4)^2}} = \mathrm{e}^{-32\sqrt{n}/\log^2 n} .$$

□

3.4.2 Martingales and the Method of Bounded Differences

Martingales are a well-studied concept in classical probability theory. Here we will define them in a discrete setting in the simplest form, which is sufficient for our purposes.

Definition 3.64 *A* martingale *is a sequence of random variables* $X_0, X_1, \ldots$ *such that*

$$\mathrm{E}[X_i \mid X_0, X_1, \ldots, X_{i-1}] = X_{i-1}$$

for all $i \geq 1$.

Let us give a simple example of a martingale. Suppose we are given a sum of independent random variables $Y = Y_1 + \ldots + Y_n$. Let $X_0 = 0$ and the random variables X_i be equal to $\sum_{j=1}^{i}(Y_i - \mathrm{E}[Y_i])$ for all $1 \leq i \leq n$. We will show that the variables X_i, $i \geq 0$, define a martingale. First, observe that they satisfy the so-called Markov property, $\mathrm{E}[X_n \mid X_0, X_1, \ldots, X_{n-1}] = \mathrm{E}[X_n \mid X_{n-1}]$, which intuitively says that the future outcome depends only on the current state. Hence,

$$\begin{aligned}
\mathrm{E}[X_n \mid X_0, X_1, \ldots, X_{n-1}] &= \mathrm{E}[X_n \mid X_{n-1}] \\
&= \mathrm{E}[X_{n-1} + (Y_n - \mathrm{E}[Y_n]) \mid X_{n-1}] \\
&= X_{n-1} + \mathrm{E}[Y_n - \mathrm{E}[Y_n] \mid X_{n-1}] \\
&= X_{n-1} + \mathrm{E}[Y_n - \mathrm{E}[Y_n]] \quad \text{by independence of the } Y_i \\
&= X_{n-1}\,, \quad \text{since } \mathrm{E}[Y_n - \mathrm{E}[Y_n]] = 0.
\end{aligned}$$

A general way of constructing martingales is given in the following definition.

Definition 3.65 *The* Doob sequence *of a function* f *with respect to a sequence of random variables* $X_1, \ldots, X_n$ *is defined by*

$$Y_i = \mathrm{E}[f \mid X_1, \ldots, X_i]$$

for all $0 \leq i \leq n$. *In particular,* $Y_0 = \mathrm{E}[f]$ *and* $Y_n = f(X_1, \ldots, X_n)$.

It is shown, for instance, in [DP98] that this sequence always defines a martingale (choose $f = \sum_{i=1}^{n} X_i$ to see why).

The following concentration result for martingales is known as Azuma's Inequality [Azu67] although it was first shown (implicitly) in a paper by Hoeffding (see [Hoe63]).

Theorem 3.66 (Azuma Inequality) *Let* $X_0, X_1, \ldots$ *be a martingale satisfying the property that* $|X_i - X_{i-1}| \leq c_i$ *for all* $i \geq 1$. *Then for any* $t \geq 0$,

$$\Pr[X_n > X_0 + t] \leq \mathrm{e}^{-t^2/(2\sum_{i=1}^{n} c_i^2)}$$

and

$$\Pr[X_n < X_0 - t] \leq \mathrm{e}^{-t^2/(2\sum_{i=1}^{n} c_i^2)}\,.$$

Using the Doob martingale, this theorem can be generalized to obtain a very powerful and useful generalization of the Chernoff bounds (see, e.g., [DP98]).

Theorem 3.67 (Method of Bounded Martingale Differences) *Let* $X_1, \ldots, X_n$ *be an arbitrary set of random variables and let* f *be a function satisfying the property that for each* $i \in [n]$, *there is a non-negative* c_i *such that*

$$|\mathrm{E}[f \mid X_1, \ldots, X_i] - \mathrm{E}[f \mid X_1, \ldots, X_{i-1}]| \leq c_i\,.$$

Then

$$\Pr[f \geq \mathrm{E}[f] + t] \leq \mathrm{e}^{-t^2/(2\sum_{i=1}^{n} c_i^2)}$$

and

$$\Pr[f \leq \mathrm{E}[f] - t] \leq \mathrm{e}^{-t^2/(2\sum_{i=1}^{n} c_i^2)}\,.$$

Note that the X_i's in this theorem are *not* required to be independent.

3.5 Witness Structures

There are certain applications where all previous theorems may not be good enough to obtain reasonable bounds. In this case it is sometimes possible to establish a connection between the performance of the algorithm and some combinatorial structure in the following way:

> *If algorithm A has a performance of P, then there exists a combinatorial structure of type $\mathcal{C}(P)$ that witnesses this.*

If it is possible to show that for certain values of P the probability of $\mathcal{C}(P)$ to exist is very low, then this immediately implies that only with a very low probability the algorithm will have a performance that matches one of these values.

There are no fixed rules according to which to construct combinatorial witness structures. We will demonstrate the approach by three examples.

3.5.1 Examples

Stability of shortest-in-system

Recall the stochastic injection model defined in Section 2.2.1. The SIS protocol (which is a short form of "shortest-in-system") always gives preference to the most recently injected packet. Ties may be broken arbitrarily. We prove the following result for SIS.

Theorem 3.68 *SIS is universally stable in the stochastic injection model.*

Proof. In order to prove the theorem, we intend to find a sequence of packets that witnesses a high delay of a packet.

Let $0 < \epsilon < 1$ be chosen such that $\lambda = 1 - \epsilon$. Suppose that there is a packet P following a path of length d that has a routing time of more than $\sum_{i=1}^{d} \tau/\epsilon^{2i}$ steps for some $\tau \in \mathbb{N}$. In this case there must exist an $i \in \{1, \ldots, d\}$ for which P was delayed for at least τ/ϵ^{2i} steps at the ith edge of its path. Consider the minimum i for which this holds. Let e be the corresponding edge on P's path. Then the time interval I from the injection of P till the time when P waited at e for the τ/ϵ^{2i}th time consists of at most

$$\begin{aligned} t &= \sum_{j=1}^{i} \tau/\epsilon^{2j} = \frac{\tau}{\epsilon^2} \cdot \frac{(1/\epsilon)^{2i} - 1}{(1/\epsilon)^2 - 1} \\ &= \tau \cdot \frac{(1/\epsilon)^{2i} - 1}{1 - \epsilon^2} \end{aligned}$$

time steps. Let the random variable X denote the number of packets generated during I with paths containing e. Since SIS is a greedy protocol and P can only be prevented to pass e by packets that are at least as young as P, it holds: If P had to wait for at least τ/ϵ^{2i} steps at e, then $X \geq \tau/\epsilon^{2i}$. Clearly,

$$\begin{aligned} \mathrm{E}[X] &\leq (1 - \epsilon) \cdot t = \frac{1}{1 + \epsilon} \cdot \tau \cdot ((1/\epsilon)^{2i} - 1) \\ &\leq \left(1 - \frac{\epsilon}{1 + \epsilon}\right) \frac{\tau}{\epsilon^{2i}} . \end{aligned}$$

This is less than the number of packets that have to delay P at e. Because of the independence assumptions in our stochastic injection model, we can apply the Chernoff bounds (see Theorem 3.35) to show that for any $0 < \epsilon \leq 1$,

$$\begin{aligned}\Pr\left[X \geq (1+\epsilon)\left(1 - \tfrac{\epsilon}{1+\epsilon}\right)\tfrac{\tau}{\epsilon^{2i}}\right] &\leq \mathrm{e}^{-\epsilon^2\cdot\left(1-\frac{\epsilon}{1+\epsilon}\right)\cdot\frac{\tau}{\epsilon^{2i}}/3} \\ &\leq \mathrm{e}^{-\epsilon^2\cdot(1/2)\cdot(\tau/\epsilon^{2i})/3} \leq \mathrm{e}^{-\tau\cdot(1/\epsilon)^{2(i-1)}/6} \ .\end{aligned}$$

Hence, the probability that the routing time of P exceeds $\sum_{i=1}^{d}\tau/\epsilon^{2i}$ is at most

$$\sum_{i=1}^{d}\mathrm{e}^{-\tau\cdot(1/\epsilon)^{2(i-1)}/6} \leq d\cdot\mathrm{e}^{-\tau/6} \ .$$

Let the random variable $t(P)$ denote the routing time of P. We know from equality (3.6) in Section 3.2.5 that

$$\mathrm{E}[t(P)] = \sum_{\tau=1}^{\infty}\Pr[t(P)\geq\tau] \leq T_0 + \sum_{\tau=T_0+1}^{\infty}\Pr[t(P)\geq\tau]$$

for any $T_0 \in \mathbb{N}$. Since

$$\sum_{i=1}^{d}(\tau+1)/\epsilon^{2i} - \sum_{i=1}^{d}\tau/\epsilon^{2i} \leq d\cdot\epsilon^{-2d}$$

for any $\tau \geq 1$, we therefore obtain with $T_0 = \sum_{i=1}^{d} 6\ln d/\epsilon^{2i}$ that

$$\begin{aligned}\mathrm{E}[t(P)] &\leq \sum_{i=1}^{d}6\ln d/\epsilon^{2i} + \sum_{\tau\geq 6\ln d}(d\cdot\epsilon^{-2d})\cdot(d\cdot\mathrm{e}^{-\tau/6}) \\ &= O\left(d\cdot(\ln d)/\epsilon^{2d} + \sum_{\tau\geq 6\ln d} d^2\cdot\epsilon^{-2d}\cdot\mathrm{e}^{-\tau/6}\right) \\ &= O\left(d\cdot(\ln d)/\epsilon^{2d}\right) \ ,\end{aligned}$$

completing the proof. □

Performance of longest-in-system

The LIS protocol (which is a short form of "longest-in-system") always gives preference to the oldest packet. We assume that there is a mechanism to break ties that prevents packets from delaying each other in a cycle (for instance, use the number of the generator that injected the packet).

A network is called *leveled of depth* L if its nodes can be arranged in at most $L+1$ sets $S_0,\ldots,S_L$ (also called *levels*) such that every edge leads from a node in set S_i to a node in set S_{i+1} for some $i \geq 0$. We will prove the following result which implies the stability of LIS. (A better result for LIS has been shown implicitly in [SV99], but it requires a more involved analysis.)

Theorem 3.69 *For any leveled network of depth L and in-degree at most Δ, and stochastic injections with injection rate at most $\lambda = 1-\epsilon$ for any $\epsilon > 0$, LIS ensures that the expected routing time of any fixed packet is at most $\Theta(\frac{\log(1/\epsilon)}{\epsilon^2}L\log\Delta)$.*

Proof. In order to prove this theorem, we use a so-called *delay sequence argument*. A delay sequence witnesses that a packet needs many time steps to reach its destination. We will show that "large" delay sequences are very unlikely, so that each packet only experiences a small delay, w.h.p. In the following, we denote by the *rank* of a packet its injection time. Thus, LIS always prefers the packet of minimum rank. A set $S \subset \mathbb{N}$ is called an *interval* if there are $a, b \in \mathbb{N}$ such that $S = \{a, a+1, \ldots, b\}$.

Definition 3.70 *Let P denote a packet, and let $s, \ell, r \geq 1$ denote integers. A (P, s, ℓ, r)-delay sequence consists of*

- *a path of ℓ* delay edges *$e_1, \ldots, e_\ell$ starting at the destination node of packet P, called* delay path*, and*
- *ℓ non-empty intervals of ranks $R_1, \ldots, R_\ell$ such that $|\bigcup_{i=1}^{\ell} R_i| = r$. The maximum rank in R_1 is equal to the injection time of P, and, for $2 \leq i \leq \ell$, the maximum rank in R_i is equal to the minimum rank in R_{i-1}.*

A (P, s, ℓ, r)-delay sequence is called active *if the stochastic generators inject s distinct packets $P_1, \ldots, P_s$ (different from P) such that, for every $1 \leq i \leq s$, packet P_i has a rank in R_j and its routing path includes edge e_j for some $1 \leq j \leq \ell$. $P_1, \ldots, P_s$ are called the* delay packets.

Using this definition, we will show the following lemma.

Lemma 3.71 *Suppose a packet P_0 takes more than $L + T$ steps to reach its destination. Then there is an active (P_0, s, ℓ, r)-delay sequence with $s \geq r + T$ and $\ell \leq L$.*

Proof. The active delay sequence is constructed incrementally. We start our construction at the time when packet P_0 reaches its destination. From this time point, we go backwards in time till we reach the first time point at which P_0 was delayed by another packet. Let the edge at which the delay happened be called e_1', and let P_1 be the packet that delayed P_0, since it was preferred against P_0 to cross e_1'. We now follow P_1 backwards in time until we reach the first edge e_2' at which P_1 was forced to wait because of a packet P_2. We change the packet again and follow P_2 backwards in time. This construction is continued until we reach a packet P_s that was not delayed in an earlier step. The path from the destination of P_0 to the source of P_s recorded by this process represents the delay path. It consists of contiguous parts of routing paths of $P_0, \ldots, P_s$. Since we have a leveled network, packets can only move from lower to higher levels. Hence, the delay path has a length ℓ of at most L, proving the last condition of Lemma 3.71.

Let the edges of the delay path be called $e_1, \ldots, e_\ell$. The rank intervals $R_1, \ldots, R_\ell$ are chosen in such a way that R_i is the interval of minimum size that includes all ranks of the packets recorded by the construction at edge e_i. In this case, $\bigcup_{i=1}^{\ell} R_i$ represents the interval from the injection time of P_s to the injection time of P_0. Thus, $r = \mathrm{rank}(P_0) - \mathrm{rank}(P_s) + 1$.

The time t covered by the construction is equal to $s + \ell$, since at each time step we either counted a delay or a move. Due to our assumption that P_0 takes more than $L + T$ steps to reach its destination, we have

$$t > L + T + (\mathrm{rank}(P_0) - \mathrm{rank}(P_s)) = L + T + (r - 1) .$$

Hence,

$$s = t - \ell \geq r + L + T - \ell \geq r + T .$$

It remains to prove that the delay sequence is active. This follows directly from our construction and the fact that the packets are preferred according to the LIS rule together with a rule for breaking ties that prevents packets from delaying each other in a cycle. □

Now consider some fixed packet P_0. Suppose that P_0 needs more than $L+T$ time steps to reach its destination. Then at least $T+r$ distinct packets are needed for an active (P_0, s, ℓ, r)-delay sequence to witness this situation. We will show that this is very unlikely to happen if T is sufficiently large.

Suppose that the delay edges and the rank intervals for the delay edges are fixed. Define R to be the set of all pairs (r, i) of integers with $1 \leq i \leq \ell$ and $r \in R_i$. For every element $(r, i) \in R$, let the binary random variable $X^g_{(r,i)}$ be one if and only if generator g injects a packet with rank r and whose path contains edge e_i. Let $X = \sum_{r,i,g} X^g_{(r,i)}$. On the other hand, let the binary random variable Y^g_r be one if and only if there is an i such that $X^g_{(r,i)} = 1$, and let $Y = \sum_{r,g} Y^g_r$. Obviously, we always have $Y \leq X$. Since any sequence of packets used as delay packets for a delay sequence must have the property that all packets are distinct and every delay packet must have a rank in R_i and must contain edge e_i in its path for some $1 \leq i \leq \ell$, Y counts the maximum number of packets that could be used as delay packets for the construction of a delay sequence. Thus, if $Y < T + r$, then the given delay path and rank intervals cannot be used for a witness that P_0 took more than $L+T$ steps to reach its destination.

Because the rank of a packet corresponds to its injection time and the expected number of packets injected in a time step that include edge e_i in their path is at most $1-\epsilon$,

$$\mathrm{E}[X] \;\leq\; \sum_{i=1}^{\ell} |R_i|(1-\epsilon) = (1-\epsilon)(r+\ell-1) \leq (1-\epsilon)s \tag{3.51}$$

for $T \geq \ell$. Thus, also $\mathrm{E}[Y] \leq (1-\epsilon)s$. Since for each generator and time step the probability of injecting a packet is independent of other generators and previous time steps, the Y^g_r's are independent. Hence, we can apply the Chernoff bounds. For $\delta = \epsilon/(1-\epsilon)$, we obtain that

$$\begin{aligned}
\Pr[Y \geq s] &= \Pr[Y \geq (1+\delta)(1-\epsilon)s] \\
&\leq \mathrm{e}^{-\min[\delta^2,\,\delta]\cdot(1-\epsilon)s/3} \\
&= \mathrm{e}^{-\min[\frac{\epsilon^2}{1-\epsilon},\,\epsilon]s/3} \leq \mathrm{e}^{-\epsilon^2 s/3}
\end{aligned}$$

since $\min[\frac{\epsilon^2}{1-\epsilon},\ \epsilon] \geq \epsilon^2$.

Up to now we have only bounded the probability that an active delay sequence exists for a fixed path and fixed rank intervals. It therefore remains to count the number of possibilities to choose a delay path and to assign rank intervals to its edges.

Let us consider some fixed values for s, ℓ, and r. The number of different (P_0, s, ℓ, r)-delay sequences can be bounded as follows. Since the given leveled network has an in-degree of at most Δ, there are at most Δ^ℓ possibilities of choosing a delay path of length ℓ that ends at the destination of P_0. Furthermore, the number of possibilities of assigning rank intervals to the edges is at most $\binom{r+\ell-1}{\ell-1}$. Thus, altogether the probability that, for some fixed s, ℓ, and r and $T \geq L$, there exists an active (P_0, s, ℓ, r)-delay sequence, is at most

$$\Delta^\ell \cdot \binom{r+\ell-1}{\ell-1} \cdot \mathrm{e}^{-\epsilon^2 s/3} \leq \Delta^\ell \cdot \binom{s}{\ell-1} \cdot \mathrm{e}^{-\epsilon^2 s/3}\,.$$

Hence, if $T = \Theta(\frac{\log(1/\epsilon)}{\epsilon^2} L \log \Delta)$ is sufficiently large, then the probability that there is a delay sequence witnessing a delay of at least $L+T$ is at most

$$\begin{aligned}
\sum_{s=T}^{\infty}\sum_{\ell=1}^{L}\sum_{r=1}^{s}\Delta^{\ell}\cdot\binom{s}{\ell-1}\cdot \mathrm{e}^{-\epsilon^2 s/3} &\leq \sum_{s=T}^{\infty} s\cdot\Delta^{L}\cdot\binom{s}{L}\cdot \mathrm{e}^{-\epsilon^2 s/3}\\
&\leq \sum_{s=T}^{\infty} s\cdot\left(\frac{\mathrm{e}s\Delta}{L}\right)^{L}\cdot \mathrm{e}^{-\epsilon^2 s/3}\\
&\leq \sum_{s=T}^{\infty} \mathrm{e}^{-\epsilon^2 c\cdot s} \quad \text{for some constant } c>0\\
&= \frac{\mathrm{e}^{-\Theta(\epsilon^2 T)}}{\epsilon^2}\,.
\end{aligned}$$

It remains to bound the expected routing time of some fixed packet P_0. Let $t(P_0)$ denote the routing time of P_0. We know from Section 3.2.5 that

$$\mathrm{E}[t(P_0)] = \sum_{\tau=1}^{\infty}\Pr[t(P_0)\geq\tau] \leq T_0 + \sum_{\tau=T_0+1}^{\infty}\Pr[t(P_0)\geq\tau]$$

for any $T_0 \in \mathbb{N}$. Choosing $T_0 = \Theta(\frac{\log(1/\epsilon)}{\epsilon^2}L\log\Delta)$ large enough, we obtain from above that

$$\begin{aligned}
\mathrm{E}[t(P_0)] &\leq T_0 + \sum_{\tau=T_0+1}^{\infty}\frac{\mathrm{e}^{-\Theta(\epsilon^2\tau)}}{\epsilon^2}\\
&= O\left(T_0 + \frac{\mathrm{e}^{-\Theta(\epsilon^2 T_0)}}{\epsilon^4}\right)\\
&= O(T_0)
\end{aligned}$$

since $\mathrm{e}^{-\Theta(\epsilon^2 T_0)} = \mathrm{e}^{-\Omega(\log(1/\epsilon))}$. This proves the theorem. □

Analysis of a simple balls-into-bins strategy

Assume that we have n balls and n bins. Each ball is placed in a bin that is chosen independently and uniformly at random. We noticed at the end of Example 2 in Section 3.2.2 that with a probability of at least $1/2$ the maximum number of balls in a bin is at least $(1-o(1))\log n/\log\log n$. In the following we demonstrate that a much better distribution of balls among the bins can be achieved if the balls have more than just one option.

Suppose that n balls are thrown sequentially into n bins in the following way: When it is the turn of ball i, 2 bins are chosen uniformly and independently at random for it. The ball is placed in the bin with the minimum number of already placed balls. Ties are broken arbitrarily. Let this placement strategy be called *minimum rule*. Then the following theorem holds. Its proof is based on a proof in [CFM+98], which itself builds on proofs in [CMM+98]. Both papers use so-called witness trees for the analysis. The idea of using witness structures for balls-into-bins strategies goes back to [KLM92]. Witness trees for balls-into-bins strategies have first been used in [Sch93].

Theorem 3.72 *If n is sufficiently large, the maximum load of a bin caused by the minimum rule is at most $4\log\log n$, w.h.p.*

Proof. We prove the theorem in two parts. First, we show that if there is a bin at the end with $4d$ balls, where $d=\log\log n$, there exists a tree structure that witnesses this situation. Afterwards, we show that, with high probability, no such witness tree exists.

Constructing a witness tree

We will consider witness trees that are rooted trees of directed edges that are labeled with numbers in $\{1, \ldots, n\}$. Each node in a witness tree represents a bin and each edge (v, w) with label i represents ball i with first choice bin v and second choice bin w. The edges can be either directed towards or away from the root.

Suppose that some bin r has a load of at least $4d$ at the end. We construct a witness tree for this situation as follows. The root of the tree corresponds to bin r. Let $b_{i_1}, \ldots, b_{i_{4d}}$ be (any $4d$ of) the balls placed in r, $i_1 < i_2 < \ldots < i_{4d}$. Let r_j be the other bin choice associated with ball b_{i_j}. For each $1 \leq j \leq 4d$, the root r is connected via a directed edge with label i_j to a node representing bin r_j. The edge direction depends on whether r or r_j is the first choice of ball b_{i_j}. Since $i_1 < i_2 < \ldots < i_{4d}$, the load of bin r before ball b_{i_j} was inserted was at least $j-1$. Therefore, the load of bin r_j, the other choice of b_{i_j}, was at least $j-1$ at that time. We use this fact to recursively grow a tree rooted at each r_j.

The witness tree we constructed in this manner is irregular. However, it contains as a subgraph a d-ary tree of depth d (that is, of $d+1$ levels) such that

- The root in level 0 has d children that are internal nodes.
- Each internal node on levels 1 to $d-2$ has two children that are internal nodes and $d-2$ children that are leaves.
- Each internal node on level $d-1$ has d children that are leaves.

For convenience we refer to this subtree as the actual witness tree henceforth.

Constructing a pruned witness tree

If the nodes of the witness tree are guaranteed to represent distinct bins, proving our probabilistic bound is a relatively easy matter. However, this is not the case: a ball may reappear several times in a witness tree, leading to dependencies that are difficult to resolve. This makes it necessary to *prune* the tree. We will show that it suffices to prune it in such a way that each *internal* node of the resulting tree represents a distinct bin. This is done as follows:

We visit the nodes of the witness tree iteratively in a breadth-first search order starting at the root and ending with the last node in level $d-1$. Assume that we reached node v and let $B(v)$ denote the set of all nodes of the possibly already pruned tree that were visited before v. If v represents a bin that is different from the bins represented by the nodes in $B(v)$, we do nothing. Otherwise, we prune all nodes in the subtree rooted at v, not including v. The procedure continues until all nodes have been visited. The tree resulting from this process is called a *pruned witness tree*.

We note two vital properties of pruned witness trees.

Lemma 3.73 *The pruned witness tree has the following properties:*

(1) All endpoints of the edges pointing to an internal node of the pruned witness tree represent distinct choices of the balls.

(2) Each ball is represented by at most two edges of the pruned witness tree, and if it has two edges, then both of them must be incident to a leaf.

Proof. We start with a proof for item (1). From the construction of a pruned witness tree it immediately follows that all internal nodes of the pruned witness tree must be distinct. Thus, there can be no choice of a ball that is represented by more than one endpoint of an edge pointing to an internal node of the pruned witness tree.

Next we prove item (2). Suppose that a ball is represented by more than two edges. Then its edges are incident to at least 3 different internal nodes. Since each ball only has two optional bins, 2 of these internal nodes must represent the same bin, which is a contradiction to the construction of a witness tree. Further, suppose that a ball is represented by two edges, where one of them connects only internal nodes. Then, again, its edges must be incident to at least 3 different internal nodes. Again, this is not possible, which concludes the proof. □

Bounding the expected number of pruned witness trees

We showed above that if there is a bin with at least $4d$ balls, then there is a pruned witness tree that fulfills the conditions of Lemma 3.73. In order to bound the probability that such a pruned witness tree exists, we intend to bound the expected number of ways of constructing a pruned witness tree and assigning balls to its edges such that

- all choices of the balls pointing to a common node in the tree represent the same bin and
- Lemma 3.73 is true.

Such an assignment is called *legal*.

To bound the expected number of pruned witness trees with a legal assignment, we distinguish between two cases:

(1) At least one tree rooted at a child of root r has not been pruned.

(2) Every tree rooted at a child of the root r has been pruned at least once.

In the first case, we restrict the witness tree to any one of the non-pruned subtrees under the root and throw away all other subtrees. This results in a d-ary witness tree of depth of $d-1$, called *of type 1*, that fulfills the conditions of Lemma 3.73. We start with counting the number of possibilities of assigning bins to the nodes and balls to the edges of a witness tree of type 1, called T_1 henceforth. Clearly, T_1 consists of $\Delta = \sum_{i=0}^{d-2} 2^i = 2^{d-1} - 1$ internal nodes and a total number of $m = d \cdot \Delta + 1$ nodes. There are n possibilities to select a bin for the root of T_1. Given any internal node v that has already been assigned to a bin, there are $\binom{n}{d}$ possibilities of choosing d balls for the edges leading to the children of v. If v does not have only leaves as children, then there are $\binom{d}{2}$ further possibilities of determining those 2 of the d balls whose edges lead to the 2 children that are internal nodes. Moreover, each edge can have 2 orientations, depending on whether the bin represented by v denotes its first or second choice. Once a ball and the number of its choice is fixed, the child of v with which its edge is connected has a uniquely determined bin (the other choice of the ball, whatever this might be). Hence, there are at most

$$
\begin{aligned}
n \cdot \left(\binom{n}{d} \binom{d}{2} 2^d \right)^{\Delta} &\leq n \cdot \left(\left(\frac{2en}{d} \right)^d \cdot \frac{d^2}{2} \right)^{\Delta} \\
&\leq n \cdot \left(\frac{4en}{d} \right)^{d \cdot \Delta} = n \cdot \left(\frac{4en}{d} \right)^{m-1}
\end{aligned}
$$

ways of assigning balls and bins to the edges and nodes of T_1. Due to Lemma 3.73 (1), all ball-choice pairs represented by the endpoints of the edges towards the root must be distinct. Since for each ball-choice pair a bin is selected uniformly and independently at random, the probability that the assignment of balls and bins to T_1 is legal is at most $n^{-(m-1)}$. Thus, the expected number of legal assignments to T_1 is at most

$$\begin{aligned} n \cdot \left(\frac{4en}{d}\right)^{m-1} \cdot \left(\frac{1}{n}\right)^{m-1} &= n \cdot \left(\frac{4e}{d}\right)^{m-1} \\ &\leq n \cdot \left(\frac{1}{2}\right)^{d(2^{d-1}-1)} \leq n^{-\log\log n/2+2} \end{aligned}$$

if n is sufficiently large.

Now we consider the second case, that is, every tree rooted at a child of the root r has been pruned at least once. In this case, for each tree rooted at a child of r we only keep its levels up to the first level that has a pruned node. Thus all nodes of this level become leaves. This results in a pruned witness tree of type 2 with root r that also fulfills Lemma 3.73. Let us omit the fact for a while that we have pruned nodes. In this case, the expected number of ways of having a legal assignment of balls to any possible witness tree of type 2 is at most

$$n \cdot \underbrace{\binom{n}{d} 2^d \left(\frac{1}{n}\right)^d}_{(*)} \underbrace{\left(\sum_{\ell=0}^{d-1} \left(\binom{n}{d}\binom{d}{2} 2^d \left(\frac{1}{n}\right)^d\right)^{2^\ell - 1}\right)^d}_{(**)} \tag{3.52}$$

where $(*)$ counts the expected number of legal assignments for the edges incident to r, and $(**)$ counts the expected number of legal assignments for the d trees rooted at the children of r, using the formula in case 1. The problem with this bound is that it is by far too weak. To improve this bound, we exploit the fact that in each subtree we have a node that caused a pruning.

Consider some fixed witness tree T_2 of type 2 with root r. Let the tree rooted at the ith child of r, $1 \leq i \leq d$, be called T_i'. For each T_i', let r_i be its root and ℓ_i be its depth. If $\ell_i = 0$, then the bin assigned to r_i must be equal to some bin assigned to r_j, $1 \leq j < i$. Since distinct balls have to be assigned to the edges incident to r and each ball chooses its bins independently at random, we obtain for each i with $\ell_i = 0$ a probability of d/n that the choice of the ball is legal. Now suppose that $\ell_i > 0$. In this case there are at most $d \cdot 2^{\ell_i}$ ways of selecting a node v_i in T_i' that caused a pruning. Let e_i be the edge incident to v_i. Every pruned node v_i has two possibilities:

(1) The ball assigned to e_i has not been assigned to some edge e_j visited earlier in the breadth-first-search order. In this case, we use the fact that the probability that the ball assigned to e_i points to a bin that has already been used somewhere else in T_2 (which has to be the case for a pruned node) is at most $d \cdot 2^{d-1}/n$.

(2) The ball assigned to e_i has already been assigned to some edge e_j visited earlier in the breadth-first-search order. According to Lemma 3.73 (2) we know that e_j must be incident to a leaf. Since the highest level in which the leaf of e_i can be is level $d-1$, the leaf of e_j must be a pruned node. If e_j is also in subtree T_i', we can treat e_i as in case (1). Otherwise, we assume w.l.o.g. that the leaf of e_j is the pruned node in the other subtree. (We can always reselect the pruned nodes accordingly.) In this case, we use the fact that there are at most d ways of choosing a ball for e_i, since only balls assigned to edges with pruned nodes can be possible candidates. We do not have to count 2 ways of orienting e_i, since only one orientation can be valid: the opposite orientation of e_j. Otherwise, Lemma 3.73 (1) is violated.

Suppose that one of the children of node v has been selected as a pruned node. Then the bound $\binom{n}{d}\binom{d}{2}2^d n^{-d}$ for v changes to

$$\binom{n}{d}2^d\left(\frac{1}{n}\right)^d \cdot \frac{d\cdot 2^{d-1}}{n}$$

for case (1) and

$$\binom{n}{d-1}2^{d-1}\left(\frac{1}{n}\right)^{d-1}\cdot d$$

for case (2). Obviously, at least $d/2$ of the pruned nodes must fulfill case (1). (Otherwise Lemma 3.73 (2) is violated.)

Since a) there are at most 2^d ways of selecting one of the two cases for each of the d pruned nodes and b) the number of ways to select a pruned node in a subtree of depth ℓ_i is at most $d\cdot 2^{\ell_i}$, the expected number of legal assignments of balls and bins to any possible T_2 is at most

$$n\cdot\binom{n}{d}2^d\left(\frac{1}{n}\right)^d\cdot\underbrace{2^d}_{a)}\cdot$$
$$\left(\underbrace{\frac{d}{n}}_{\ell_i=0}+\sum_{\ell=1}^{d-1}\underbrace{d\cdot 2^{\ell_i}}_{b)}\cdot\left(\binom{n}{d}\binom{d}{2}2^d\left(\frac{1}{n}\right)^d\right)^{2^\ell-2}\cdot\underbrace{\binom{n}{d}2^d\left(\frac{1}{n}\right)^d\cdot\frac{d\cdot 2^{d-1}}{n}}_{(1)}\right)^{d/2}\cdot$$
$$\left(\underbrace{\frac{d}{n}}_{\ell_i=0}+\sum_{\ell=1}^{d-1}\underbrace{d\cdot 2^{\ell_i}}_{b)}\cdot\left(\binom{n}{d}\binom{d}{2}2^d\left(\frac{1}{n}\right)^d\right)^{2^\ell-2}\cdot\underbrace{\binom{n}{d-1}2^{d-1}\left(\frac{1}{n}\right)^{d-1}\cdot d}_{(2)}\right)^{d/2}.$$

If n is large enough, then $\binom{n}{d}4^d n^{-d}\le 1$ and $\binom{n}{d-1}2^{d-1}n^{d-1}d\le 1$. This simplifies the bound above to at most

$$\begin{aligned} & n\cdot\left(\frac{d}{n}+d(2^d-1)\cdot\frac{d\cdot 2^{d-1}}{n}\right)^{d/2}\left(\frac{d}{n}+d(2^d-1)\right)^{d/2}\\ \le\ & n\cdot\left(\frac{(d2^d)^2}{n}\right)^{d/2}\cdot\left(d2^d\right)^{d/2} = n\cdot\left(\frac{(d2^d)^3}{n}\right)^{d/2}\\ = n^{-\Omega(\log\log n)}\ . \end{aligned}$$

From our construction above it follows that if there is a bin with at least $4d$ balls, then there either exists a type 1 witness tree, or a type 2 witness tree. Let the random variable X denote the number of witness trees of type 1 and 2 that can be constructed from the given choices of the balls. According to our calculations above it follows that

$$\mathrm{E}[X] = n^{-\Omega(\log\log n)}\ .$$

If there exists a bin with at least $4d$ balls, then we must have $X\ge 1$. However, according to the Markov inequality,

$$\Pr[X\ge 1]\le \mathrm{E}[X] = n^{-\Omega(\log\log n)}\ .$$

This proves the theorem. □

3.6 The Lovász Local Lemma

Probabilistic methods have been used with great success to prove the existence of objects with desirable properties by showing that a randomly chosen object from an appropriate probability distribution has the desired properties with positive probability. In most applications this probability is not only positive, but is actually high and frequently tends to 1 as the parameters of the problem tend to infinity. In such cases, the proof usually supplies an efficient randomized algorithm for producing a structure of the desired type.

There are, however, certain examples, where one can prove the existence of the required combinatorial structure by probabilistic arguments that deal with rare events; events that hold with positive probability which is exponentially small in the size of the input. This happens often when using the Lovász Local Lemma. We present this lemma in its general form.

A directed graph $G = (V, E)$ on the set of nodes $V = \{1, 2, \ldots, n\}$ is called a *dependency graph* for the events $A_1, \ldots, A_n$ if for every i, $1 \le i \le n$, the event A_i is independent of any set of events A_j with $(i, j) \notin E$.

Lemma 3.74 (Lovász Local Lemma [EL75]) *Let $A_1, \ldots, A_n$ be "bad" events in an arbitrary probability space. Suppose that $G = (V, E)$ is a dependency graph of these events and suppose there are real numbers $x_i \in [0, 1)$ for all $1 \le i \le n$ with*

$$\Pr[A_i] \le x_i \prod_{(i,j)\in E} (1 - x_j)$$

for all $1 \le i \le n$. Then,

$$\Pr[\bigcap_{i=1}^{n} \bar{A}_i] \ge \prod_{i=1}^{n} (1 - x_i) \ .$$

In particular, with positive probability no bad event A_i holds.

Proof. We first prove, by induction on s, that for any set $S \subset \{1, \ldots, n\}$, $|S| = s$, and $i \notin S$,

$$\Pr[A_i \mid \bigcap_{j\in S} \bar{A}_j] \le x_i \ . \tag{3.53}$$

This is certainly true for $s = 0$. Assuming it holds for all $s' < s$, we prove it for S. Set $S_1 = \{j \in S : \ (i, j) \in E\}$ and $S_2 = S \setminus S_1$. Then

$$\Pr[A_i \mid \bigcap_{j\in S} \bar{A}_j] = \frac{\Pr[A_i \cap (\bigcap_{j\in S_1} \bar{A}_j) \mid \bigcap_{\ell\in S_2} \bar{A}_\ell]}{\Pr[\bigcap_{j\in S_1} \bar{A}_j \mid \bigcap_{\ell\in S_2} \bar{A}_\ell]} \ . \tag{3.54}$$

To bound the numerator, observe that since A_i is mutually independent of the events $\{A_\ell : \ \ell \in S_2\}$,

$$\begin{aligned}\Pr[A_i \cap (\bigcap_{j\in S_1} \bar{A}_j) \mid \bigcap_{\ell\in S_2} \bar{A}_\ell] &\le \Pr[A_i \mid \bigcap_{\ell\in S_2} \bar{A}_\ell] \\ &= \Pr[A_i] \le x_i \prod_{(i,j)\in E} (1 - x_j) \ . \end{aligned} \tag{3.55}$$

The denominator, on the other hand, can be bounded by the induction hypothesis. Indeed, suppose $S_1 = \{j_1, j_2, \ldots, j_r\}$. If $r = 0$, then the denominator is 1, and (3.53) follows. Otherwise

$$\begin{aligned}\Pr[\bigcap_{j\in S_1} \bar{A}_j \mid \bigcap_{\ell\in S_2} \bar{A}_\ell] &= \prod_{k=1}^{r} \left(1 - \Pr[A_{j_k} \mid \bar{A}_{j_1} \cap \ldots \cap \bar{A}_{j_{k-1}} \cap (\bigcap_{\ell\in S_2} \bar{A}_\ell)]\right) \\ &\ge \prod_{k=1}^{r} (1 - x_{j_k}) \ge \prod_{(i,j)\in E} (1 - x_j) \ . \end{aligned} \tag{3.56}$$

Substituting (3.55) and (3.56) into (3.54), we conclude that $\Pr[A_i \mid \bigcap_{j \in S} \bar{A}_j] \leq x_i$, completing the proof of the induction.

The assertion of Lemma 3.74 now follows easily, as

$$\begin{aligned} \Pr[\bigcap_{i=1}^{n} \bar{A}_i] &= \prod_{i=1}^{n} \left(1 - \Pr[A_i \mid \bigcap_{j=1}^{i-1} \bar{A}_j]\right) \\ &\geq \prod_{i=1}^{n} (1 - x_i) \ . \end{aligned}$$

□

Lemma 3.75 (The Local Lemma, Symmetric Case) *Let $A_1, \ldots, A_n$ be events in an arbitrary probability space. Suppose that each event A_i is mutually independent of a set of all the other events A_j but at most d, and that $\Pr[A_i] \leq p$ for all $1 \leq i \leq n$. If*

$$\mathrm{e}p(d+1) \leq 1 \tag{3.57}$$

then $\Pr[\bigcap_{i=1}^{n} \bar{A}_i] > 0$.

Proof. If $d = 0$, the result is trivial. Otherwise, by the assumption, there is a dependency graph $G = (V, E)$ for the events $A_1, \ldots, A_n$ in which for each i, $|\{j : (i,j) \in E\}| \leq d$. The result now follows from Lemma 3.74 by taking $x_i = 1/(d+1) < 1$ for all i and using the fact that for any $d \geq 1$, $[1 - 1/(d+1)]^d > 1/\mathrm{e}$. □

It is worth noting that, as was shown by Shearer [She85], the constant "e" is the best possible constant in inequality (3.57). In the following, we present various applications of the Local Lemma. There is no known proof of any of these results that does not use the Local Lemma.

3.6.1 Existential Proofs using the Local Lemma

First, we present several existential results using the Local Lemma. We start with two important, elementary problems.

Hypergraph coloring

Consider the problem of coloring the nodes of a hypergraph $H = (V, E)$. H is called *c-colorable* if c colors are sufficient to color its nodes in such a way that no hyperedge is *monochromatic*, i.e., only has nodes of a single color. We prove a sufficient condition for the 2-colorability of hypergraphs.

Theorem 3.76 ([EL75]) *Let $H = (V, E)$ be a hypergraph in which every edge has at least k elements, and suppose that every edge of H intersects at most d other edges. If $\mathrm{e}(d+1) \leq 2^{k-1}$, then H is 2-colorable.*

Proof. Color each node v of H randomly and independently, either blue or red with equal probability. For each edge $e \in E$, let A_e be the event that e is monochromatic. Obviously, $\Pr[A_e] = 2/2^{|e|} \leq 1/2^{k-1}$. Moreover, each event A_e is clearly mutually independent of all the events $A_{e'}$ for which e' does not intersect e. The result now follows from Lemma 3.75. □

Set partitioning

The *set partitioning problem* [FK86] is defined as follows:

Let $H = (V, E)$ be a set system with $V = [n]$ and $E = \{S_1, \dots, S_M\} \subseteq \mathcal{P}(V)$. Given a positive integer ℓ, the problem is to partition V into ℓ parts, so that the sets are split well: we want a $\chi : V \to [\ell]$ that minimizes $Y = \max_{j\in[M],k\in[\ell]} |\{i \in S_j : \chi(i) = k\}|$.

We prove the following result.

Theorem 3.77 *Let $H = (V, E)$ be a system of sets of size k, and suppose that every set in H intersects at most d other sets. For any $\ell \in \mathbb{N}$ it holds that if*

$$\mathrm{e} \cdot \ell(d + 1) \leq \mathrm{e}^{\frac{\epsilon^2 k}{2\ell(1+\epsilon/3)}} ,$$

then there is a set partitioning for H with $Y \leq (1 + \epsilon)k/\ell$.

Proof. Consider the random experiment of choosing independently and uniformly at random one out of ℓ colors for each node. For each set $S \in E$, let A_S denote the event that S has a set of more than $(1 + \epsilon)k/\ell$ nodes of the same color. For some fixed $S \in E$, let X_i denote the number of nodes in S with color i, $1 \leq i \leq \ell$. Obviously, $\mathrm{E}[X_i] = k/\ell$. Since the nodes choose their colors independently of the other nodes, we can use the Chernoff bounds to obtain

$$\Pr[X_i \geq (1 + \epsilon)\mathrm{E}[X_i]] \leq \mathrm{e}^{-\frac{\epsilon^2 k/\ell}{2(1+\epsilon/3)}} .$$

From this it follows that

$$\Pr[A_S] \leq \ell \cdot \mathrm{e}^{-\frac{\epsilon^2 k/\ell}{2(1+\epsilon/3)}} .$$

The theorem now follows directly from Lemma 3.75. □

Routing

Next we will present a more advanced application of the Local Lemma that requires to use the set partitioning problem several times. Consider the problem of routing packets in an arbitrary collection of simple paths with congestion C and dilation D, one packet along each path. The proof of the following result has been taken from [Sch98].

Theorem 3.78 ([LMR94]) *For any collection of paths with congestion C and dilation D, there is an offline schedule that needs at most $O(C + D)$ time steps to send a packet along each of these paths.*

Proof. The global framework of our strategy for constructing an efficient schedule is the same as in [LMR94]. It consists of a succession of refinements, starting with an initial schedule S_0, in which each packet is forwarded at every time step until it reaches its final destination. This initial schedule is as short as possible. Its length is exactly D. Unfortunately, up to C packets may have to use an edge at a single time step in S_0, whereas in the final schedule at most one packet is allowed to use an edge at each step. Each refinement will bring us closer to meeting this requirement by bounding the congestion within smaller and smaller time intervals. The proof uses the Chernoff bounds and the Lovász Local Lemma at each refinement step. In the following, a *t-interval* denotes a time interval consisting of t consecutive time steps. A *t-frame* is defined as a t-interval that starts at an integer multiple of t.

Let $I_0 = \max[C, D]$ and $I_j = \max[\log I_{j-1},\ 6]$ for all $j \geq 1$. The first step is to assign an initial delay to each packet, chosen independently and uniformly at random from the range $\Delta_1 = [C]$. In the resulting schedule, S_1, a packet that is assigned a delay of δ waits in its source node for δ steps, then moves on without waiting again until it reaches its destination. The length of S_1 is at most $D + C$. We use the Lovász Local Lemma to show that if the delays are chosen independently and uniformly at random and I_1 is sufficiently large, then with nonzero probability the congestion at any edge in any I_1^3-interval is at most $(1 + \frac{4}{I_1})I_1^3$. Thus, such a set of delays must exist.

To apply the Lovász Local Lemma, we associate a bad event with each edge. The bad event for edge e is that more than $C_1 = (1 + \frac{4}{I_1})I_1^3$ packets use e in some I_1^3-interval. For this we have to bound the dependence b among the bad events and the probability p that a bad event occurs.

We first bound the dependence. Whether or not a bad event occurs depends solely on the delays assigned to the packets that pass through the corresponding edge. Thus, two bad events are independent unless some packet passes through both of the corresponding edges. Since at most C packets pass through an edge and each of these packets passes through at most D other edges, the dependence b of the bad events is at most $C \cdot D$.

Next we bound the probability that a bad event occurs. Let us consider some fixed edge e and I_1^3-interval J. Let the packets traversing e be numbered from 1 to $m \leq C$. For every $i \in \{1, \ldots, m\}$, let the binary random variable X_i be one if and only if packet i traverses e during J. Let $X = \sum_{i=1}^{m} X_i$. Since the packets choose their delays uniformly at random from a range of size C, we get $\Pr[X_i = 1] \leq I_1^3/C$ for all $i \in \{1, \ldots, m\}$. Hence,

$$\mathrm{E}[X] = \sum_{i=1}^{m} \mathrm{E}[X_i] \leq I_1^3 \ .$$

Together with the Chernoff bounds we therefore get with $\epsilon = \frac{4}{I_1}$ (which is at most 1 because of $I_1 \geq 6$) that

$$\Pr[X \geq C_1] = \Pr[X \geq (1+\epsilon)I_1^3] \leq \mathrm{e}^{-\left(\frac{4}{I_1}\right)^2 I_1^3/3} \leq 2^{-6I_1} \ .$$

Since for each edge there are at most $D + C$ many I_1^3-intervals to consider, the probability that a bad event occurs for edge e is bounded by

$$p \leq (D + C)2^{-6I_1} < \frac{1}{\mathrm{e}(CD+1)}$$

for all $I_1 \geq 6$ (assuming that $C, D \geq 6$). Thus, the product $\mathrm{e}p(b+1)$ is less than 1 and therefore, by the Lovász Local Lemma, there is some assignment of delays such that the congestion in each I_1^3-interval is bounded by $(1 + \frac{4}{I_1})I_1^3$.

We now break schedule S_1 into I_1^4-frames and continue to schedule each frame independently. So each frame can be viewed as a separate scheduling problem where the origin of a packet is its location at the beginning of the frame, and its destination is its location at the end of the frame. Our next refinement step will be to choose, for each frame, a random initial delay for each packet that visits at least one edge within this frame. In the resulting schedule S_2, the frames (enlarged by their delay ranges) are executed one after the other in a way that a packet that is assigned a delay δ in some frame F waits at its first edge in F for (additional) δ steps, and then moves on without waiting until it traverses its last edge in F.

Let us concentrate in the following on some fixed frame F. Let each packet choose an additional delay out of the range $\Delta_2 = [I_1^3 - I_2^3]$. Hence the length of the resulting schedule for this frame is at most $I_1^4 + I_1^3$. We use the Lovász Local Lemma to show that if the delays are chosen independently and uniformly at random and I_2 is sufficiently large, then with nonzero probability the congestion in any I_2^3-interval is at most

$$C_2 = \left(1 + \frac{4}{I_1}\right)\left(1 + \frac{6}{I_2}\right)\left(\frac{1}{1 - (I_2/I_1)^3}\right) I_2^3 \ .$$

We again associate a bad event with each edge. The bad event for edge e is that more than C_2 packets use e in some I_2^3-interval.

We first bound the dependence of these events. Whether or not a bad event occurs solely depends on the delays assigned to the packets that pass through the corresponding edge. Since at most $C_1 \cdot I_1$ packets pass through an edge in any frame and each of these packets passes through at most I_1^4 other edges, the dependence of the bad events is at most $C_1 \cdot I_1^5$.

Next we bound the probability that a bad event occurs. Let us consider some fixed edge e and I_2^3-interval J. We know according to the first refinement step that every I_1^3-interval has congestion at most C_1. Since the packets can only choose delays up to $I_1^3 - I_2^3$ and J covers I_2^3 time steps, at most C_1 packets are able to traverse this edge during J. Let these packets be numbered from 1 to $m \le C_1$. For every $i \in \{1, \ldots, m\}$, let the binary random variable X_i be one if and only if packet i traverses e during J. Let $X = \sum_{i=1}^m X_i$. Since the packets choose their delays uniformly at random from a range of size $I_1^3 - I_2^3$, we get $\Pr[X_i = 1] \le I_2^3/(I_1^3 - I_2^3)$ for all $i \in \{1, \ldots, m\}$. Hence,

$$\mathrm{E}[X] = \sum_{i=1}^m \mathrm{E}[X_i] \le \left(1 + \frac{4}{I_1}\right)\left(\frac{1}{1 - (I_2/I_1)^3}\right) I_2^3 \ .$$

Together with the Chernoff bounds we therefore get with $\epsilon = \frac{6}{I_2}$ and $\alpha_2 = (1 + \frac{4}{I_1})(\frac{1}{1-(I_2/I_1)^3})$ that

$$\Pr[X \ge C_2] = \Pr[X \ge (1+\epsilon)\alpha_2 I_2^3] \le \mathrm{e}^{-\left(\frac{6}{I_2}\right)^2 \alpha_2 I_2^3/3} \le \mathrm{e}^{-12\alpha_2 I_2} \ .$$

Since for each edge there are at most $I_1^4 + I_1^3$ many I_2^3-intervals to consider, the probability that a bad event occurs for edge e is bounded by

$$p \le (I_1^4 + I_1^3)\mathrm{e}^{-12\alpha_2 I_2} < \frac{1}{\mathrm{e}(C_1 I_1^5 + 1)}$$

for all $I_2 \ge 6$. Thus, the product $\mathrm{e}p(b+1)$ is less than 1 and therefore, by the Lovász Local Lemma, there is some assignment of delays such that the congestion in each I_2^3-interval is bounded by C_2.

We continue to refine each I_2^4-frame in F recursively until we reach a round k, in which $I_k = 6$. We end up with a schedule S_k with total length at most

$$(D + C)\prod_{j=1}^{k-1}\left(1 + \frac{1}{I_j}\right) = O(D + C)$$

and with a congestion C_k in each I_k^3-interval of at most

$$I_k^3\left(1 + \frac{4}{I_1}\right)\prod_{j=1}^{k-1}\left(1 + \frac{6}{I_{j+1}}\right)\left(\frac{1}{1 - (I_{j+1}/I_j)^3}\right) = O(I_k^3) \ .$$

In order to obtain a feasible schedule, we simply schedule each I_k^3-frame in schedule S_k in a greedy way. Since each I_k^3-frame has a constant congestion and dilation, we altogether end up with a routing schedule of runtime $O(C + D)$. □

3.6.2 Constructive Proofs using the Local Lemma

Many applications of the Local Lemma can be found in the literature (see, e.g., [ABLP91, Alo94, AMR91, ASE92, BFU97, EL75, ES91, FS98, HMR97, LMR94, Mol98, Ree98, RS98, Spe95, Sri96]). To turn proofs using the Lovász Local Lemma into efficient algorithms, even random ones, proved to be difficult for many of these applications. In a breakthrough paper [Bec91], Beck presented a method of converting some applications of the Lovász Local Lemma into polynomial time algorithms (with some sacrifices made with regards to the constants in the original application). Alon [Alo91] provided a parallel variant of the algorithm and simplified the arguments used. Leighton, Maggs and Richa [LMR99] extended Beck's arguments in 1996 so that they can also be applied to scheduling problems. Similar techniques were later used by Molloy and Reed [MR98] to find polynomial time algorithms for a general class of applications of the Lovász Local Lemma.

In the following, we will concentrate on the presentation and extension of the methods given by Alon and Molloy and Reed. The first result constructivizes Theorem 3.76.

An algorithm for 2-coloring hypergraphs

Theorem 3.79 ([Alo91]) *Let $H = (V, E)$ be a hypergraph in which every edge has at least k elements, and every edge intersects less than d other edges, where k and d are fixed. Suppose that for some $\alpha > 0$*

$$4\mathrm{e}d^3 \leq 2^{(1-H(\alpha))k} \tag{3.58}$$

and

$$2\mathrm{e}d \leq 2^{\alpha k} \ . \tag{3.59}$$

Then there is a randomized algorithm that finds a 2-coloring for H in polynomial time.

Proof. Suppose that we want to find a 2-coloring for a hypergraph H consisting of n nodes and m edges, labeled e_1 to e_m. The algorithm for finding a proper 2-coloring of H consists of several passes. In the first pass, all nodes of H are colored randomly and independently by one out of two colors, where each point is colored either red or blue with equal probability. We call an edge *bad* if at most αn of its points are red or at most αn of its points are blue. According to Section 3.1.2, the probability of a fixed edge to be bad is at most

$$2 \sum_{i \leq \alpha k} \binom{k}{i} 2^{-k} \leq 2 \cdot 2^{(H(\alpha)-1)k} \ ,$$

where $H(\alpha)$ is the binary entropy function. Put $p = 2 \cdot 2^{(H(\alpha)-1)k}$ and let B denote the set of all bad edges.

Let G be the dependency graph for the problem, whose nodes are the hyperedges of H and in which two nodes are connected if and only if their corresponding hyperedges intersect. Observe that if S is an independent set in G, then the probability that $S \subset B$ is at most $p^{|S|}$, since these $|S|$ events are mutually independent. Let us call a set of nodes C of G a 1,2-tree if C is the set of nodes of a connected subgraph in the square of G, i.e., C is a 1,2-tree if the nodes in C are such that drawing an edge between two nodes in C if their distance in G is either 1 or 2 the resulting graph is connected.

Lemma 3.80 *The probability that every 1,2-tree in G all of whose nodes belong to B has a size of at most $d\log(2m)$ is at least $1/2$.*

Proof. We call $T \subseteq G$ a 2,3-tree if the nodes in T are such that all their mutual distances in G are at least two and so that, drawing an edge between two nodes in T if their distance in G is either 2 or 3 the resulting graph is connected. We first bound the number of 2,3-trees of size s in G. Consider the graph on the set of nodes of G in which two nodes are adjacent if their distance in G is either 2 or 3. Every 2,3-tree on a set T of nodes of G must contain a tree on T in this new graph. The new graph has a maximum degree of at most $D = (d-1)^3 + (d-1)^2 < d^3$. It is well known (see, e.g., [Knu69], p. 369 (Exercise 11)) that an infinite D-regular rooted tree contains precisely $\frac{1}{(D-1)s+1}\binom{Ds}{s}$ rooted subtrees of size s, and this easily implies that the number of trees of size s containing one specific given node in any graph with maximum degree at most D does not exceed this number, which is smaller than $(\mathrm{e}D)^s$.

For any particular 2,3-tree T of size s we know that $\Pr[T \subseteq B] \leq p^s$. Hence, the expected number of 2,3-trees $T \subseteq B$ of size s is at most $m(\mathrm{e}Dp)^s$. As $\mathrm{e}Dp \leq 1/2$ by (3.58), if $s = \log(2m)$ this term is at most $1/2$. Thus with probability at least $1/2$ there is no 2,3-tree of size bigger than $\log(2m)$ all of whose nodes are in B. We actually want to bound the size of any 1,2-tree C of G all of whose nodes lie in B. A maximal 2,3-tree T in such a C must have the property that every node in C is a neighbor (in G) of a node in T. This is because otherwise there is a node in C which is not a neighbor of any node in T, and yet it is of distance 2 or 3 from some node in T. Such a node can be added to T, contradicting its maximality. Since there are less than d neighbors of any node in G, $\log(2m) \geq |T| \geq |C|/d$ and so

$$|C| \leq d\log(2m)$$

completing the proof of the lemma. □

Let us call the first pass successful if there is no 1,2-tree of size greater than $d\log(2m)$ all of whose nodes lie in B. By the last lemma, the probability that the first pass is successful is at least $1/2$. In case it is not, we simply repeat the entire procedure. In expected linear time the first pass is successful.

We can now fix the coloring by recoloring, in the second pass, the nodes of H that belong to the bad edges. Let us call an edge *dangerous* if it contains at least αk nodes that belong to bad edges. (Thus, in particular, bad edges are also dangerous.) Observe that if an edge is not dangerous then it will not become monochromatic after the recoloring. This is because less than αk of its points will change color, and it has at least αk points of each color before the recoloring. Thus we only have to worry about the dangerous edges. However, if we recolor all the nodes in the bad edges randomly and independently, then we recolor at least αk nodes in each dangerous edge, and hence the probability that it becomes monochromatic does not exceed $2^{-\alpha k+1}$. Since each dangerous edge intersects less than d others, it follows from (3.59) and Theorem 3.76 that there is a recoloring in which no edge is monochromatic.

The crucial point is that the recoloring of the points in the edges of each maximal 1,2-tree C of bad edges can be done separately. This is because there is no dangerous edge that intersects edges from two distinct such maximal 1,2-trees, and hence it suffices to recolor the points in the edges of each such C in a way that only makes sure that no dangerous edge intersecting an edge in C becomes monochromatic. Since we assume d and k to be fixed, each of the 1,2-trees C as above has only $O(\log m)$ nodes that have to be recolored. Thus we can find the required recoloring by exhaustive enumeration, as examining all possible two-colorings in each such C only takes time polynomial in m. □

Extending the 2-coloring algorithm to non-constant d and k

If d and k are not fixed, in order to obtain a polynomial time algorithm, one can make another pass similar to the first one in each 1,2-tree separately, which results in new 1,2-trees of size $O(d \log(d \log m))$. However, since this may reduce the size of the hyperedges up to some α^2 fraction of their original size, we have to change the conditions for α in Theorem 3.79 to

$$4\mathrm{e}d^3 \leq 2^{(1-H(\alpha))\alpha k} \quad \text{and} \quad 2\mathrm{e}d \leq 2^{\alpha^2 k} .$$

to ensure that a second reduction is possible. At this point, we distinguish between two cases. Let c be a sufficiently large constant. If $k \leq c \log\log\log n$ then $d = (\log\log n)^{O(1)}$. Thus, we can simply use exhaustive search in each 1,2-tree to obtain a polynomial time algorithm. Otherwise, we perform another random experiment of choosing a color independently and uniformly for each node covered by a 1,2-tree. Since there are at most $O(d \log(d \log m))$ hyperedges in a 1,2-tree we have: If c is large enough, with probability at least $1/2$ a coloring will be chosen in a 1,2-tree such that all edges are monochromatic. Thus, we also end up with a polynomial time algorithm in this case.

A general method for the symmetric LLL

In this section, we present a technique developed by Molloy and Reed [MR98] that seems to capture all applications to which Beck's technique will apply.

In the following, let $\mathcal{T} = \{t_1, \ldots, t_n\}$ be a set of independent random trials, and let $\mathcal{A} = \{A_1, \ldots, A_m\}$ be a set of events such that each A_i is determined by the outcome of the trials in $T_i \subseteq \mathcal{T}$. We say that T_i *intersects* T_j and A_i *intersects* A_j ($A_i \sim A_j$) if $T_i \cap T_j \neq \emptyset$.

For any $t_{j_1}, \ldots, t_{j_k} \in T_i$ and any $w_{j_1}, \ldots, w_{j_k}$ in the domains of $t_{j_1}, \ldots, t_{j_k}$, we define $\Pr^*[A_i \mid t_{j_1} = w_{j_1}, \ldots, t_{j_k} = w_{j_k}]$ to be the probability of A_i conditional on the event that the outcomes of $t_{j_1}, \ldots, t_{j_k}$ are $w_{j_1}, \ldots, w_{j_k}$. We sometimes just say $\Pr^*[A_i]$ if it causes no ambiguity, always meaning $t_{j_1}, \ldots, t_{j_k}$ to be the set of trials already carried out and $w_{j_1}, \ldots, w_{j_k}$ to be their outcomes. For $k = 0$, we have $\Pr^*[A_i] = \Pr[A_i]$.

Theorem 3.81 ([MR98]) *If we have the following:*

(1) for each $1 \leq i \leq m$, $\Pr[A_i] \leq p$;

(2) each T_i intersects less than d other T_j's;

(3) $p \cdot d^9 \leq (1/2\mathrm{e})^3$;

(4) for each $1 \leq i \leq m$, $|T_i| \leq w$;

(5) for each $1 \leq j \leq n$, the size of the domain of t_i is at most γ, and we can carry out the random trial in time τ_1;

(6) for each $1 \leq i \leq m$, $t_{j_1}, \ldots t_{j_k} \in T_i$ and $w_{j_1}, \ldots, w_{j_k}$ in the domains of $t_{j_1}, \ldots t_{j_k}$, we can compute $\Pr^*[A_i]$ in time τ_2;

then we have a randomized $O(n \cdot d \cdot (\tau_1 + \tau_2) + n \cdot \gamma^{wd(\log(d \log m)+2)})$-time algorithm which will find outcomes for $t_1, \ldots, t_n$ such that none of the events in $\mathcal{A}$ holds.

The algorithm and the proof are essentially the same as those in [Alo91] and [Bec91]. The new idea here is to consider the conditional probabilities $\Pr^*[A_i]$ and to occasionally undo a trial.

In the first pass, we carry out the trials $t_1, \ldots, t_n$ in sequential order. After each trial t_j, we compute $\Pr^*[A_i]$ for every i with $t_j \in T_i$. If $\Pr^*[A_i] > p^{2/3}$, then we say that A_i is *dangerous*, and we

- undo t_j, i.e., we cancel its outcome and carry out the trial again at a later time, and
- freeze t_j and all other remaining trials in T_i, i.e., we will not carry out those trials during this pass and so when it comes to their turn we will skip them.

Note that at the end of the first pass, $\Pr^*[A_i] \leq p^{2/3}$ for all i. Therefore, by the (symmetric form of the) Local Lemma, upon carrying out the remaining trials,

$$\Pr[\bar{A}_1 \cap \ldots \cap \bar{A}_m] > 0 .$$

That is, there is a feasible solution extending the partial solution given by the trials that have already been carried out. We will see that so few of the events become dangerous that it is now nearly feasible to find a good set of outcomes for the remaining trials using exhaustive search.

Lemma 3.82 *For each $1 \leq i \leq m$, the probability that A_i becomes dangerous is at most $p^{1/3}$.*

Proof. If the probability that $\Pr^*[A_i]$ will ever exceed $p^{2/3}$ is greater than $p^{1/3}$, then $\Pr[A_i] > p$. □

We denote by H the hypergraph with node set $V = \mathcal{T}$ and edge set $E = \{T_1, \ldots, T_m\}$ representing the events. Let a 1,2-tree and a 2,3-tree of hyperedges in H be defined as in Section 3.6.2. We call an a, b-tree *dangerous* if all of its hyperedges correspond to dangerous events.

The key observation is this: No A_i intersects two events which belong to different maximal dangerous 1,2-trees. Thus, we can deal with the frozen trials contained in each maximal dangerous 1,2-tree independently, and so as long as these trees is sufficiently small, an exhaustive search for each tree of all the combinations of possible outcomes of the corresponding trials is feasible.

Lemma 3.83 *With probability at least $1/2$, all dangerous 1,2-trees have a size of at most $d\log(2m)$.*

Proof. The proof is similar to Lemma 3.80. We require that $\mathrm{e}d^3p^{1/3} \leq 1/2$ for the lemma to hold. This is fulfilled due to condition (3) above. □

If after the first pass we have dangerous 1,2-trees of size greater than $d\log(2m)$, then we repeat the first pass. The expected number of repetitions is constant, and each repetition takes time $O(n \cdot d \cdot (\tau_1 + \tau_2))$. At this point, an exhaustive search for the satisfactory outcomes to the frozen trials corresponding to each dangerous 1,2-tree takes time $O(\gamma^{wd\log(2m)})$. Thus, if d, γ, and w are constant, this can be done in polynomial time. To improve the runtime, we run a second pass in the same manner as the first pass, where we carry out the frozen trials in sequence, and an event becomes dangerous if its conditional probability exceeds $p^{1/3}$. Within an expected linear number of repetitions of the second pass, there will be no dangerous 1,2-trees of size greater than $d\log(2m')$ where $m' = d\log(2m)$, and so we can complete our exhaustive search of each one in time $O(\gamma^{wd(\log(d\log m)+2)})$. Thus the total expected running time is

$$O\left(n \cdot d \cdot (\tau_1 + \tau_2) + n \cdot \gamma^{wd(\log(d\log m)+2)}\right) .$$

An improved general method for the symmetric LLL

Now we show how to improve the algorithm by Molloy and Reed to obtain polynomial time algorithms for (almost) all applications that match their conditions.

Theorem 3.84 *If we have the following:*

(1) for each $1 \le i \le m$, $\Pr[A_i] \le p$;

(2) each T_i *intersects less than* d *other* T_j*'s;*

(3) for each $1 \le i \le m$, $|T_i| \le w$;

(4a) $p \cdot d^9 \le (1/2\mathrm{e})^3$;

(4b) $p \cdot w \le 1$; *(this is the only additional condition!)*

(5) for each $1 \le j \le n$, *we can carry out the random trial in time* τ_1;

(6) for each $1 \le i \le m$, $t_{j_1}, \ldots t_{j_k} \in T_i$ *and* $w_{j_1}, \ldots, w_{j_k}$ *in the domains of* $t_{j_1}, \ldots t_{j_k}$*, we can compute* $\Pr^*[A_i]$ *in time* τ_2;

then we have a randomized $O(n \cdot d \cdot (\tau_1 + \tau_2) + n(\tau_1 \cdot \log^{O(1)} m + 2^{(\log\log m)^{O(1)}}))$*-time algorithm which will find outcomes for* $t_1, \ldots, t_n$ *such that none of the events in* $\mathcal{A}$ *holds.*

Proof. In the following, we always assume that m is sufficiently large. The first two passes work in the same way as in the algorithm by Molloy and Reed. However, instead of using exhaustive search at the end, we distinguish between the following two cases:

If $p^{1/3} \le 1/(2d\log(2d\log(2m)))$, then we simply carry out all the random trials in each 1,2-tree T. Since the probability of a dangerous event to be true is ensured to be at most $p^{1/3}$ and every 1,2-tree consists of at most $d\log(2d\log(2m))$ dangerous events, the probability that there is an event left in T that is true is at most $1/2$. Hence, in an expected number of $O(\tau_1 \cdot n)$ steps all events are made false.

Suppose otherwise that $p^{1/3} > 1/(2d\log(2d\log(2m)))$. According to condition (4a) it holds that $d^3 \le 1/(2\mathrm{e}p^{1/3})$. Thus, if m is sufficiently large, this restricts d to be at most $\log\log m$. From this we obtain together with condition (4b) that

$$w \le 1/p \le (2d\log(2d\log(2m)))^3 \le (3\log\log m)^6 .$$

Thus, altogether $d \cdot w \le 3^6(\log\log m)^7$. If $\gamma = \log^{O(1)} m$, then we use exhaustive search to find outcomes for the trials so that all events are false. This takes at most $2^{(\log\log m)^{O(1)}}$ time for T. If $\gamma = \log^{\Omega(1)} m$, we will first use a strategy presented below that reduces γ to a value in $\log^{O(1)} m$. Afterwards, we continue as for $\gamma = \log^{O(1)} m$.

In order to reduce γ from $\log^{\Omega(1)} m$ to $\log^{O(1)} m$, we use the following simple sampling algorithm for each 1,2-tree T:

For each random trial t_j in T, perform s random experiments (s will be in the order of $\log^{O(1)} m$). The outcomes of these random experiments are stored in vector $\boldsymbol{v}_j = (v_{j,1}, \ldots, v_{j,s})$.

We show that if we change now the random experiment in a way that each t_j is determined by choosing an entry in $\boldsymbol{v}_j$ independently and uniformly at random, then we still have a probability of at most $2p^{1/3}$ for a dangerous event to become true, w.h.p. In this case, condition (4a) implies that according to the symmetric LLL there must be outcomes for the trials from the sample sets such that all events are false. (If this is not true, because the sampling failed to guarantee a probability of at most $2p^{1/3}$ for a dangerous event to become true, we will notice it after the exhaustive search and simply restart the sampling procedure above.)

Lemma 3.85 *Consider some fixed 1,2-tree T. If $s = \log^{O(1)} m$, then with probability at least $1 - 1/m$ outcome vectors are chosen such that every dangerous event in T has a probability of at most $2p^{1/3}$ to be true.*

Proof. We will use Theorem 3.52 to prove the lemma. Consider any fixed event A in T, and let T_A denote the set of its trials, $|T_A| = q$. W.l.o.g. we assume that $T_A = \{t_1, \ldots, t_q\}$. Let $M = [s^q]$ represent the set of all possibilities of choosing an outcome vector $O = (v_{1,j_1}, \ldots, v_{1,j_q})$ out of the sample vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_q$. Suppose that we choose uniformly at random a subset $U \subset M$ of size k, representing the outcome vectors $O_\ell = (v_{1,j_{1,\ell}}, \ldots, v_{q,j_{q,\ell}})$, $1 \le \ell \le k$. Let the binary random variable X_ℓ be 1 if and only O_ℓ implies that A is true. Our aim is to bound

$$z_\ell = \Pr[X_\ell = 1 \mid X_1 = \ldots = X_{\ell-1} = 1]$$

for all $1 \le \ell \le k$. Let the event E_ℓ be true if and only if outcome O_ℓ has at least one entry in common with an outcome $O_{\ell'}$ with $1 \le \ell' < \ell$, that is, there is an $\ell' \in [\ell - 1]$ and an $i \in [q]$ such that $j_{i,\ell} = j_{i,\ell'}$. Clearly,

$$\Pr[E_\ell] \le \binom{q(\ell-1)}{1} \frac{s^{q-1}}{s^q} = \frac{q \cdot (\ell - 1)}{s} .$$

On the other hand, we know that

$$\Pr[X_\ell = 1 \mid X_1 = \ldots = X_{\ell-1} = 1 \ \wedge \ \bar{E}_\ell] \le p^{1/3} .$$

Thus, altogether we have

$$\begin{aligned} z_\ell &= \Pr[E_\ell] \cdot \Pr[X_\ell = 1 \mid X_1 = \ldots = X_{\ell-1} = 1 \ \wedge \ E_\ell] + \\ &\quad\ \Pr[\bar{E}_\ell] \cdot \Pr[X_\ell = 1 \mid X_1 = \ldots = X_{\ell-1} = 1 \ \wedge \ \bar{E}_\ell] \\ &\le \Pr[E_\ell] + \Pr[X_\ell = 1 \mid X_1 = \ldots = X_{\ell-1} = 1 \ \wedge \ \bar{E}_\ell] \\ &\le \frac{q \cdot k}{s} + p^{1/3} . \end{aligned}$$

Let us choose $s = q \cdot k \cdot (\log m)/p^{1/3}$. From above we know that $q \le (3 \log\log m)^6$. Furthermore, we intend to choose $k = \alpha \log m$ for some constant $\alpha > 0$. Thus, $s = (\log m)^{O(1)}$. Moreover, we obtain

$$z_\ell \le \left(1 + \tfrac{1}{\log m}\right) p^{1/3}$$

for all $1 \le \ell \le k$.

For any $i \in M$, let the binary random variable Y_i be 1 if and only if A is true for outcome vector i, and let $Y = \sum_{i \in M} Y_i$. Obviously, it holds that $\Pr[Y_i = 1] \le p^{1/3}$. Thus,

$$\mathrm{E}[Y] \le s^q \cdot p^{1/3} .$$

From above we know that, for the multilinear polynomial $P_j(.)$ used in the proof of Theorem 3.52, it holds that

$$P_j(Y_1, \ldots, Y_{s^q}) \le P_j(p', \ldots, p') \quad \text{with } p' = (1 + \tfrac{1}{\log m}) p^{1/3}$$

for all $1 \le j \le k$. Thus, according to Theorem 3.52, we have

$$\Pr[Y \ge 2p^{1/3}] \le \mathrm{e}^{-\min[(1/2)s^q p^{1/3}/3,\ \lfloor k/2 \rfloor]} \le \frac{1}{m^2}$$

if $k = \alpha \log m$ and α is a sufficiently large constant. Hence, the probability of A to become true (if each t_j in A is determined by choosing an entry in $\boldsymbol{v}_j$ independently and uniformly at random) is at most $2p^{1/3}$ with probability at least $1 - 1/m^2$. Since T has less than m dangerous events, the lemma follows. $\square$

Lemma 3.85 concludes the proof of Theorem 3.84. $\square$

Chapter 4

Transforming Static into Dynamic Switching Protocols

In this chapter, we investigate how static online switching protocols can be transformed into efficient dynamic online switching protocols, that is, how algorithms that have been designed for the case that all packets are injected at the same time can be adapted to more realistic scenarios in which packets are continuously injected into the network. We will present a black box transformation scheme applicable to every static switching protocol. In order to analyze the performance of the transformed protocols, we use stochastic and adversarial models of packet injections.

Our black box transformation ensures that if the static online algorithm is pure (i.e., no extra packets apart from the original packets are sent), its dynamic variant is stable up to a maximum possible injection rate. Furthermore, in the stochastic model, the routing time of a packet depends on local parameters such as the length of its routing path, rather than on the maximum possible path length, even if the static algorithm chosen for the transformation does not provide this locality feature and is not pure. In the latter case, however, the dynamic variant may not be stable any more up to a maximum possible injection rate. In the adversarial model, the delay bound of the packets is closely related to the time bound given for the static algorithm.

4.1 Introduction

In the static switching model we are given a network and a set of packets specified as paths in this network. The task of a switching protocol is to send these packets along their selected paths. Many static switching protocols have been developed in recent years (see, e.g., [LMR94, LMRR94, MV95, RT96, CMSV96, OR97]). In practice, however, networks are rarely used in this static fashion but packets are injected dynamically into the network. Since much less is known in the area of dynamic switching than in the area of static switching, it has been considered as highly desirable to transfer the results gathered for static switching to dynamic switching.

In this chapter, we investigate how static switching algorithms can be transformed into dynamic switching algorithms that are stable and efficient under a stochastic or adversarial model of packet injections. We will present a simple and elegant scheme that transforms any static protocol into an efficient dynamic protocol that is stable up to a maximum possible injection rate if the protocol is *pure*, i.e., no control packets or copies of packets are allowed. Besides showing the stability of these protocols, we will prove bounds on the routing time of the packets. Furthermore, we prove that the transformed protocols recover quickly from any worst case scenario (concerning, for instance, the expected delay of a packet).

4.1.1 Models and Problems

As defined in Section 2.2, we model a network as an undirected graph, and the switching in this network is performed in a synchronous "store and forward" manner, that is, in every step, each edge can be crossed by at most one packet in each direction.

We will concentrate on switching protocols in which the nodes *locally* decide which packets to move forward in each step, i.e., a decision only depends on the routing information carried by the packets and on the local history of the execution. Recall that these algorithms are called *online* algorithms. We use the following models.

Static packet switching

Here we assume that a fixed collection of paths is given with *congestion* C and *dilation* D, where C denotes the maximum number of paths crossing an edge, and D denotes the maximum length of a path. Along each of these paths a packet has to be sent. Furthermore, we denote by M the *complexity* of the switching problem. M is defined to be the maximum of the number of edges, the number of paths, and the dilation. If we use the term "w.h.p.", we mean with a probability of at least $1 - M^{-\alpha}$, where $\alpha > 0$ is an arbitrary constant. Let us give some examples of known results about the routing time required for static packet switching:

- $C \cdot D$: trivial upper bound for any greedy protocol in case of unlimited buffers, i.e., protocols in which a packet is only delayed because other packets move along the next edge on the packet's routing path;
- $(1 + \kappa) \cdot C + O(D \cdot \log M)$, w.h.p., for any constant $\kappa > 0$: upper bound for arbitrary paths in arbitrary networks with unbounded buffers [LMR94].
- $O(C + D + \log M)$, w.h.p.: upper bound in leveled networks with buffers of constant size, where D is the depth of the network [Ran91, LMRR94], and upper bound for shortest paths in arbitrary networks with unbounded buffers [MV99, MV95];
- $(1 + \kappa) \cdot C + (\log^* M)^{O(\log^* M)} D + O((\log M)^6)$, w.h.p., for any constant $\kappa > 0$: upper bound for simple paths in arbitrary networks with unbounded buffers [RT96];
- $O(C + D + \log^{1+\kappa} M)$, w.h.p., for any constant $\kappa > 0$: upper bound for simple paths in arbitrary networks with unbounded buffers [OR97].

We will come back to some of these results when using our black box transformations.

Dynamic packet switching

The most commonly used injection models in the dynamic setting are the *stochastic* and the *adversarial* injection model. We use the same models as defined in Section 2.2.

Recall that for both injection models, a switching protocol is called *stable* for a given injection rate λ if the (expected) number of packets in the injection and edge buffers does not grow to infinity with time. We are interested in protocols that are stable for high injection rates. Of course, since an edge can transport at most one packet per step, λ can be at most 1. Our aim is to construct algorithms that are stable under injection rates that are close to 1. Additionally, we are interested in short delays for the packets, i.e., we aim to minimize the time from injection to service for each packet.

Apart from the stability and the routing time we will consider another property of dynamic switching protocols, the recovery from worst case scenarios. Although our bounds on the routing time guarantee that bad configurations are very unlikely, they eventually occur from time to time when the switching protocol runs for an infinite number of time steps. Let a *worst case scenario* denote an arbitrarily bad configuration of the network. Then the *recovery time* with regard to some property $\mathcal{P}$ of the switching protocol is defined as the time that has to pass by after the occurrence of a worst case scenario until $\mathcal{P}$ holds again. In our case, we are interested in properties such as the expected routing time of a packet and time bounds that hold w.h.p.

Similar to the static model we define the complexity of the dynamic switching problem to be a value capturing all relevant parameters. In particular, the complexity is defined to be the maximum of the number of edges, the number of generators, the maximum possible length of a routing path, and $1/(1-\lambda)$. (The number of generators in the adversarial model is defined to be w times the number of edges, which corresponds to the maximum number of packets that can be injected in a single step.)

In the following sections, we will prove results for both the stochastic and the adversarial injection model.

4.1.2 Previous Results

Not many results are known about how to transform static switching protocols into dynamic switching protocols. To our knowledge, the first results about adaptations of static, universal switching protocols to the dynamic setting for the stochastic injection model can be found in [SV96].

In the last couple of years a new model called *adversarial queuing theory* emerged. This approach was introduced by Borodin et al. in [BKR$^+$96]. Most research in this model focuses on the stability of switching protocols and networks. For example, Borodin et al. [BKR$^+$96] show several stability results for greedy protocols on DAGs and directed cycles. Andrews et al. [AAF$^+$96] extend their results by showing that there exist simple greedy protocols (such as longest-in-system, shortest-in-system and farthest-to-go) that are stable against any adversary for all networks. However, the delay of the packets and the number of packets stored in a queue might get exponential in the length of the longest path.

Furthermore, Andrews et al. [AAF$^+$96] present a transformation of the static protocol presented in [LMR94] into a dynamic protocol that is stable for any injection rate < 1 and fulfills the following constraint on the buffer size: for any fixed time step t, at most $(D \cdot \log m)^k$ packets are stored in any queue at time t, w.h.p., where D denotes the longest routing path, m the number of edges, and k is a suitable constant. This result implies that the delay of the packets is bounded by $D \cdot (D \cdot \log m)^k$, w.h.p.

Rabani and Tardos [RT96] present a transformation scheme which yields much better routing times. Assuming there is a static algorithm that delivers all packets in $(1+\kappa)C + g(M)D + f(M)$ steps for some constant $\kappa > 0$, their transformation yields a dynamic algorithm that delivers each packet to its destination, w.h.p., in $O(w + g(N)D + f(N) + \log N)$ against an adversary of rate $(w, \Theta(\kappa))$, where M and N denote the complexity of the static and dynamic switching problems, respectively. The stability of the dynamic algorithms, however, is not shown. In fact, although most packets will be delivered fast, some packets will never reach their destination and queues will grow to infinity assuming the stochastic model or the adversarial model using a randomized, static algorithm.

Broder et al. [BFU96] introduce a general approach to dynamic packet switching with bounded buffers in the stochastic and adversarial model. They show sufficient conditions for the stability of dynamic packet switching algorithms and investigate how some well-known static switching protocols for the butterfly network can be transformed into dynamic algorithms that fulfill these conditions. In particular, they present a dynamic switching algorithm for the butterfly that is stable for a small constant injection rate, and they show that the expected routing time for each packet is $O(\log n)$, with n denoting the number of nodes on a level.

Andrews et al. [AFH+97] investigate another, more restrictive dynamic switching model. In contrast to the stochastic and the adversarial model, the packets are injected regularly in "sessions". For each session i, packets are injected at a rate r_i to follow a fixed path of length d_i. They describe a schedule that delivers each packet in a time depending only on local parameters, that is, each packet reaches its destination in time $O(d_i + 1/r_i)$, which is worst case optimal. We will see that similar local properties, i.e., the routing time depends on d_i, can be achieved also in the stochastic model.

4.1.3 New Results

In this chapter, we present a powerful black box transformation scheme applicable to every static online switching protocol.

In the following, N denotes the complexity of the dynamic switching problem (as defined in Section 4.1.1), D denotes the maximum length of a routing path, and $\epsilon = 1 - \lambda$, where λ denotes the injection rate.

For our black box transformations, we basically combine the ideas of Rabani and Tardos [RT96] for the fast delivery of packets with the universal stability of the shortest-in-system protocol originally shown by Andrews et al. [AAF+96] for the adversarial model. The major problem that we solve is merging these two approaches so that we obtain dynamic protocols that are stable up to some injection rate depending on the static protocol without any slowdown due to the inefficiency of the shortest-in-system protocol (see Section 3.5.1).

Let $\mathcal{S}$ denote an arbitrary set of paths, for instance, the set of all simple or all shortest paths in the given network. Suppose we are given a static switching algorithm that routes all packets in $\gamma \cdot C + \delta \cdot D + O(\log^\alpha M)$ steps, with high probability (or even with certainty), for any collection of paths or subpaths in $\mathcal{S}$ with congestion C, dilation D, and complexity M. Assume that in the dynamic setting only paths in $\mathcal{S}$ are allowed to be generated. Then our black box transformation yields a dynamic variant of this protocol with the following properties. (In the following overview, we only describe the results for the case that γ and δ do not depend on C or D and $\alpha \geq 1$ is a constant. Similar results will be shown for other choices of α, γ, and δ.)

- If the given static protocol is *pure* (i.e., no control packets or copies of packets are allowed), the dynamic algorithm is stable for any injection rate $\lambda < 1$. Otherwise, it is stable for any injection rate $\lambda < 1/\gamma$.

- If $\lambda < 1/\gamma$ then the algorithm guarantees that any packet p that has to travel a distance of d is delivered in time $O(\delta \cdot d + \log^\alpha N)$, w.h.p., in the stochastic injection model, and in time $O(\delta \cdot D + w + \log^\alpha N)$, w.h.p., in the adversarial injection model.

- The algorithm recovers from any worst case scenario in $O(\delta \cdot D + w + \log^\alpha N)$ time steps (with $w = 0$ in the stochastic model).

The bound on the routing time implies that it might be important for static switching protocols to know the exact factor γ in front of the C, since this can be decisive for the performance of their dynamic counterparts. Interestingly, in the stochastic injection model, the dynamic variant is able to exploit locality, whereas the static algorithm does not need to provide this feature. For example, the transformation of a well-known static switching algorithm (see, e.g., [LMR94]) that delivers all packets in time $(1+\kappa)\cdot C+O(D\cdot\log M)$, w.h.p., for any constant $\kappa > 0$, yields a dynamic algorithm that delivers each packet p in time $O(d \cdot \log N)$, w.h.p., for any constant injection rate $\lambda < 1/(1+\kappa)$, where d is the length of p's path.

In Section 4.3, we finally show how to adapt these results to the adversarial model.

4.2 A Universal Black Box Transformation Scheme

In this section, we present a black box transformation scheme for arbitrary static switching protocols, using the stochastic injection model. In Section 4.3, we will show how the results for the stochastic model can be adapted to the adversarial model. Recall that M denotes the complexity of the static switching problem and N denotes the complexity of the dynamic switching problem.

Theorem 4.1 *Let $\mathcal{S}$ denote any set of paths in an arbitrary network. Consider any static switching protocol $\mathcal{P}$ that sends packets along every collection of paths or subpaths from $\mathcal{S}$ in at most $\gamma C + \delta D + O(\log^\alpha M)$ time steps, w.h.p. Then $\mathcal{P}$ can be transformed into a dynamic switching protocol $\mathcal{P}'$ possessing the following properties in the stochastic model:*

Suppose packets are injected at rate λ and are to be routed solely along paths or subpaths from $\mathcal{S}$. Then the expected routing time of a packet following a path of length d is at most

1) $O\left(\epsilon^{-2}\left(\delta d + \log^\alpha N\right)\right)$, *if* $\gamma = \Theta(1)$, $\delta = \Theta(\log^\beta N)$ *for some constant* $0 \leq \beta \leq 1$, *and* $\lambda = (1-\epsilon)/\gamma$ *for some* $\epsilon > 0$;

2) $O\left((\epsilon/2)^{-\frac{2}{1-\beta}}\left(d \log^{\alpha\beta} N + \log^\alpha N\right)\right)$, *if* $\gamma = \Theta(1)$, $\delta = C^\beta$ *for some constant* $0 < \beta < 1$, *and* $\lambda = (1-\epsilon)/\gamma$ *for some* $\epsilon > 0$;

3) $O\left(\epsilon^{-2}\left(d \log N + \log^2 N\right)\right)$ *if the bound on the runtime of* $\mathcal{P}$ *is* $C \cdot D$ *and* $\lambda = (1-\epsilon)/\log N$ *for some* $\epsilon > 0$.

All time bounds also hold w.h.p. Furthermore, the recovery time in all three cases is equal to the respective time bound with d replaced by $D(\mathcal{S})$, the length of the longest path in $\mathcal{S}$. If $\mathcal{P}$ is pure, then $\mathcal{P}'$ is stable for any $\lambda < 1$.

Observe that the bounds on the expected routing time of a packet imply stability up to the specified injection rate, which depends on the parameters of the static protocol. The stability of pure protocols, however, is independent of these parameters, although the delay of a packet might become exponential in its path length if the injection rate is too high.

Certainly, more than the three cases stated in the theorem can be solved with the techniques below, but we believe that these cases are the most natural ones. Case 1 covers the case in which γ, δ, and α are constants, which we believe is the most important case. Let us apply Theorem 4.1 to some of the static protocols mentioned in Section 4.1.1.

- **Switching along simple paths in arbitrary networks:** The static protocol of Ostrovsky and Rabani [OR97] with runtime $O(C + D + \log^{1+\kappa} M)$, w.h.p., for any constant $\kappa > 0$, can be transformed into a dynamic protocol that guarantees a routing time of $O(d + \log^{1+\kappa} N)$, w.h.p., for any λ up to some constant < 1.

- **Switching along arbitrary paths in arbitrary networks:** The simple static protocol with runtime $(1+\kappa) \cdot C + O(D \log M)$, w.h.p., for any constant $\kappa > 0$, presented by Leighton, Maggs, and Rao in [LMR94], can be used to obtain a dynamic protocol that guarantees a routing time of $O(d \log N)$, w.h.p., for any constant $\lambda < 1$.

- **Greedy switching along simple paths in arbitrary networks:** Any greedy switching protocol can be transformed into a dynamic protocol that delivers each packet in time $O\left(d \log N + \log^2 N\right)$, w.h.p., for any constant $\lambda < 1/\log N$. We point out that the dynamic protocol is greedy, too.

The first and the second example follow from case 1 of the theorem, whereas the last example follows from case 3. Since all of these protocols are pure, their dynamic counterparts are stable for any $\lambda < 1$.

In the following we show how to transform $\mathcal{P}$ into a dynamic protocol $\mathcal{P}'$ so that Theorem 4.1 holds.

4.2.1 Description of $\mathcal{P}'$

Let $\mathcal{P}$ be any static switching protocol, and let the injection rate λ be $1-\epsilon$ for some $\epsilon > 0$. Consider the time to be partitioned into consecutive intervals of length T. Let c_T and d_T denote suitable integers. (These parameters will be specified later.) Every newly injected packet waits until the beginning of the next T-interval. Afterwards it tries to traverse d_T edges of its path in each T-interval until it reaches its destination. Whenever it manages to traverse d_T edges within a T-interval, it waits for the next T-interval. If it fails to traverse d_T edges in some T-interval, it is declared a *failed* packet for the rest of the routing. $\mathcal{P}'$ now works as follows:

At the beginning of each T-interval, $\mathcal{P}$ is started with parameters c_T, used as a bound for the congestion, and d_T, used as a bound for the dilation. All packets that have not failed yet are allowed to participate in $\mathcal{P}$. $(1-\epsilon^2/2)T$ time steps of the T-interval are reserved for the execution of $\mathcal{P}$. (All packets that do not manage to traverse d_T edges within this time fail.) The remaining $T \cdot \epsilon^2/2$ time steps (namely, every $2/\epsilon^2$th time step of T) are reserved for the failed packets. To the failed packets, a contention resolution rule called shortest-in-system (or SIS) is applied. Recall that SIS always gives precedence to the packet most recently injected (i.e., of youngest age) in case that several packets contend for the same edge.

If $\mathcal{P}$ is pure, then we can improve $\mathcal{P}'$ so that it becomes greedy, i.e., a packet only has to wait because the next edge on its path is used by another packet. In the pure case, we use SIS as a primary contention resolution rule, but we manipulate the age of the non-failed packets as follows. As before, a packet that has not traversed $i \cdot d_T$ edges by the end of its ith T-interval is declared a *failed* packet for the rest of its life. Every newly injected packet gets an initial age of ∞ until the beginning of the next T-interval. Afterwards, as long as it has not failed or reached its destination yet, its age is set to 0 at the beginning of each T-interval. Whenever it traverses $i \cdot d_T$ edges before the end of the ith T-interval in which it participates, its age is set back to ∞ for the rest of that interval.

The age of a failed packet is defined by its injection time. In each time interval, the packets with age 0 are scheduled according to protocol $\mathcal{P}$. Note that packets that participate in $\mathcal{P}$ always have a lower age than the other packets. Hence, if we use SIS as primary contention resolution rule, we can allow the failed packets to be routed together with the other packets without disturbing the schedule of $\mathcal{P}$.

In order to analyze the performance of $\mathcal{P}'$, we need to study the behavior of SIS.

Bounding the routing time for shortest-in-system.

In this section we bound the routing time for shortest-in-system, given the following model:

Suppose that packets are injected with rate λ, using our standard stochastic model. A packet is said to be *old* if the difference between the actual time step and its injection time is more than K. Otherwise it is called *young*. As long as a packet is young, it is allowed to set its age to an arbitrary value in each step. The age of an old packet is determined by its injection time. The probability that a packet becomes old before it reaches its destination is assumed to be at most p_f (which denotes its *failure probability*). The event that a packet is old may influence the probability that another packet may also be old. We model these dependencies via a *dependency graph* $G = (V, E)$. Each node V represents a (generator,time step)-pair. Edges are chosen in G such that for any independent set $\{(g_1, t_1), \ldots, (g_k, t_k)\} \in V$ with $k \in \mathbb{N}$, under the assumption that g_i injects a packet P_i at step t_i, the probability that P_i becomes old for all $i \in \{1, \ldots, k\}$ is at most p_f^k. That is, for a proof of an upper bound on the number of old packets, any independent set of (g, t)-pairs can be viewed as having independent failure probabilities.

For this model we show the following lemma.

Lemma 4.2 *For any $0 < \epsilon < 1$ and $K \geq 0$ the SIS protocol ensures that, under the above model with injection rate $\lambda = 1 - \epsilon$, failure probability p_f, and a dependency graph G of maximum degree b, the expected routing time of a packet following a path of length d is at most*

(1) $O((d/\epsilon) \cdot (d(\epsilon^{-1} + b^2) + K))$ if $p_f \leq \epsilon/(2(4d + 1))$, and

(2) $O((K + d) \cdot d/\epsilon^{2d})$ otherwise.

Proof. We start with proving item 2. Suppose that there is a packet P that has a routing time of more than $\sum_{i=1}^{d} \tau/\epsilon^{2i}$ steps, for some $\tau \geq 2K$. In this case there must exist an $i \in \{1, \ldots, d\}$ for which P was delayed for at least τ/ϵ^{2i} steps at the ith edge of its path. Consider the minimum i for which this holds. Let e be the ith edge on P's path. Then the time interval I from the injection of P till the time when P waited at e for the τ/ϵ^{2i}th time consists of at most

$$\begin{aligned} t &= \sum_{j=1}^{i} \tau/\epsilon^{2j} = \tau \cdot \frac{1}{\epsilon^2} \cdot \frac{(1/\epsilon)^{2i} - 1}{(1/\epsilon)^2 - 1} \\ &= \tau \cdot \frac{(1/\epsilon)^{2i} - 1}{1 - \epsilon^2} \end{aligned}$$

time steps. Let the random variable X denote the number of (young and old) packets generated during I with paths containing e. Since any packet is allowed to choose its age in an arbitrary way only during the first K steps of its life it holds: if P had to wait for at least τ/ϵ^{2i} steps at e, then only the first K of these delays may be due to older packets, that is, $X \geq \tau/\epsilon^{2i} - K$. Clearly,

$$\begin{aligned} \mathrm{E}[X] &\leq (1 - \epsilon) \cdot t = \frac{1}{1 + \epsilon} \cdot \tau((1/\epsilon)^{2i} - 1) \\ &\leq \left(1 - \frac{\epsilon}{1 + \epsilon}\right) (\tau/\epsilon^{2i} - K) \ . \end{aligned}$$

This is less than the number of packets generated in I that have to delay P at e. Because of the independence assumptions in our stochastic injection model, we can apply the Chernoff bounds (see Theorem 3.35) to show that for any $0 < \epsilon \leq 1$ the probability that p is delayed by at least τ/ϵ^{2i} packets at e is at most

$$\begin{aligned}
\Pr\left[X \geq (1+\epsilon)\left(1-\tfrac{\epsilon}{1+\epsilon}\right)(\tau/\epsilon^{2i}-K)\right] &\leq \mathrm{e}^{-\epsilon^2\cdot\left(1-\frac{\epsilon}{1+\epsilon}\right)\cdot(\tau/\epsilon^{2i}-K)/3} \\
&\leq \mathrm{e}^{-\epsilon^2\cdot(1/2)\cdot(\tau-K)(1/\epsilon^{2i})/3} \leq \mathrm{e}^{-\tau\cdot(1/\epsilon)^{2(i-1)}/12} \ .
\end{aligned}$$

Hence, the probability that the routing time of P exceeds $\sum_{i=1}^{d}\tau/\epsilon^{2i}$ is at most

$$\sum_{i=1}^{d}\mathrm{e}^{-\tau\cdot(1/\epsilon)^{2(i-1)}/12} \quad\leq\quad d\cdot\mathrm{e}^{-\tau/12} \ .$$

Thus, recalling the proof of Theorem 3.68, the expected routing time of P is at most

$$\begin{aligned}
&\sum_{i=1}^{d}2K/\epsilon^{2i} + \sum_{\tau\geq 2K}(d\cdot\epsilon^{-2d})\cdot(d\cdot\mathrm{e}^{-\tau/12}) \\
&\quad\leq\ d\cdot 2K/\epsilon^{2d} + \sum_{\tau\geq 2K} d^2\cdot\epsilon^{-2d}\cdot\mathrm{e}^{-\tau/12} \\
&\quad=\ O\left((K+d)\cdot d/\epsilon^{2d}\right) \ .
\end{aligned}$$

Next we prove item 1. Suppose that there is a packet P that has a routing time of more than $K + d\cdot 4\tau/\epsilon$ steps, for some $\tau \geq K$. In this case there must exist an $i \in \{1,\ldots,d\}$ for which P was delayed for at least $4\tau/\epsilon$ time steps at the ith edge of its path while it was already old. Let I be a time interval covering $4\tau/\epsilon$ of these steps. Furthermore, let the random variable X denote the number of young packets that delayed P in I, and let the random variable Y denote the number of old packets that delayed P in I. It clearly holds that

$$\Pr[X+Y\geq 4\tau/\epsilon] \leq \Pr[X\geq\tau(4/\epsilon-1)] + \Pr[Y\geq\tau] \ .$$

We first bound the probability that $X \geq \tau(4/\epsilon - 1)$. Since any young packet contributing to X must have been injected in a time interval of size $|I| + K$,

$$\mathrm{E}[X] \leq \lambda(|I|+K) = (1-\epsilon)(4\tau/\epsilon+K) \ .$$

Because of the independence assumptions in our stochastic injection model it follows that

$$\begin{aligned}
\Pr[X\geq(1+\tfrac{\epsilon}{2(1-\epsilon)})(1-\epsilon)(4\tau/\epsilon+K)] &\leq \mathrm{e}^{-\min[(\frac{\epsilon}{2(1-\epsilon)})^2,\frac{\epsilon}{2(1-\epsilon)}](1-\epsilon)(4\tau/\epsilon+K)/3} \\
&\leq \mathrm{e}^{-\min[\frac{\epsilon}{1-\epsilon},\,2]\cdot\tau/3} \\
&\leq \mathrm{e}^{-\epsilon\cdot\tau/3} \ .
\end{aligned}$$

Since $(1+\frac{\epsilon}{2(1-\epsilon)})(1-\epsilon) = 1-\epsilon/2$ and

$$\frac{4\tau}{\epsilon} - (1-\tfrac{\epsilon}{2})\left(\frac{4\tau}{\epsilon}+K\right) \geq 2\tau - K \geq \tau \ ,$$

we also have

$$\Pr[X\geq\tau(4/\epsilon-1)] \leq \mathrm{e}^{-\epsilon\tau/3} \ .$$

Next we bound the probability that $Y \geq \tau$. Since P can only be delayed in I by old packets that are younger than P and P has an age at the end of I of at most $K + d\cdot 4\tau/\epsilon$ steps, we get with $p_f \leq \epsilon/(2(4d+1))$ that

$$\mathrm{E}[Y] \leq \left(K + \frac{d\cdot 4\tau}{\epsilon}\right)\cdot p_f \leq \frac{\tau}{2} \ .$$

In order to prove a tail estimate for Y, we apply Theorem 3.50. Using this theorem with $p = p_f$, $d = b$, and $\delta = 1/3$ we get

$$\begin{aligned}\Pr[Y \geq \tau] &\leq 4b \cdot \mathrm{e}^{-\frac{\delta^2(\tau/2)}{4b(1+\delta/3)}} \\ &= 4b \cdot \mathrm{e}^{-\tau/(80b)} .\end{aligned}$$

The probability bounds for X and Y and the fact that there are d possibilities to select an edge where P experiences a high delay imply that the probability that the routing time of P exceeds $K + d \cdot 4\tau/\epsilon$ is at most

$$d\left(\mathrm{e}^{-\epsilon\tau/3} + 4b \cdot \mathrm{e}^{-\tau/(80b)}\right) .$$

Thus, the expected routing time of P is at most

$$\begin{aligned}&(K + d \cdot 4K/\epsilon) + \\ &\sum_{\tau \geq K} [(K + d \cdot 4(\tau+1)/\epsilon) - (K + d \cdot 4\tau/\epsilon)] \cdot d\left(\mathrm{e}^{-\epsilon\tau/3} + 4b \cdot \mathrm{e}^{-\tau/(80b)}\right) \\ \leq\ & d \cdot 5K/\epsilon + \sum_{\tau \geq K} (4d/\epsilon) \cdot d \cdot \mathrm{e}^{-\epsilon\tau/3} + \sum_{\tau \geq K} (4d/\epsilon) \cdot d \cdot 4b \cdot \mathrm{e}^{-\tau/(80b)} \\ =\ & O\left(\frac{K \cdot d}{\epsilon} + \frac{d^2}{\epsilon^2} + \frac{d^2 \cdot b^2}{\epsilon}\right) = O\left(\tfrac{d}{\epsilon}(d(\tfrac{1}{\epsilon} + b^2) + K)\right) .\end{aligned}$$

□

4.2.2 Analysis of $\mathcal{P}'$ for Non-Pure $\mathcal{P}$

First let us introduce some notation. Given a time interval I, $|I|$ denotes its *size*, i.e., the time range it includes. A packet p is called a $\mathcal{P}$*-packet* in a T-interval I if $\mathcal{P}$ is applied to p in I, i.e., p was generated before I and has not failed nor reached its destination yet.

Bounding the routing time of successful packets.

To prove bounds on the routing time of successful packets, we show the following lemma.

Lemma 4.3 *Let ϕ be any constant greater than 0. Consider any static switching protocol $\mathcal{P}$ that sends packets along any path collection (consisting of paths from some specified set) with congestion C and dilation D in at most $\gamma C + \delta D + O(\log^\alpha N)$ time steps, with probability at least $1 - N^{-\phi}/2$, where $\alpha, \gamma, \delta \geq 1$. Then it holds for each T-interval I:*

(1) If $\gamma = \Theta(1)$, $\delta = \Theta(\log^\beta N)$ for some constant $0 \leq \beta \leq 1$, $\lambda = (1-\epsilon)/\gamma$ for some $\epsilon > 0$, $d_T = (\log N)^{\alpha-\beta}$, and $T = \Theta(\epsilon^{-2}(\phi \log N + \log^\alpha N))$ is sufficiently large, then with probability at least $1 - N^{-\phi}$, $\mathcal{P}'$ requires at most $(1-\epsilon^2/2)T$ time steps in I to send any fixed $\mathcal{P}$-packet along d_T edges.

(2) If $\gamma = \Theta(1)$, $\delta = C^\beta$ for some constant $0 < \beta < 1$, $\lambda = (1-\epsilon)/\gamma$ for some $\epsilon > 0$, $d_T = (\log N)^{\alpha(1-\beta)}$, and $T = \Theta((\epsilon/2)^{-\frac{2}{1-\beta}}(\phi \log N + \log^\alpha N))$ is sufficiently large, then with probability at least $1 - N^{-\phi}$, $\mathcal{P}'$ requires at most $(1-\epsilon^2/2)T$ time steps in I to send any fixed $\mathcal{P}$-packet along d_T edges.

(3) If the runtime of $\mathcal{P}$ is at most $C \cdot D$, $\lambda = (1-\epsilon)/\log N$ for some $\epsilon > 0$, $d_T = \log N$, and $T = \Theta(\epsilon^{-2}\phi \log^2 N)$ is sufficiently large, then with probability at least $1 - N^{-\phi}$, $\mathcal{P}'$ requires at most $(1-\epsilon^2/2)T$ time steps in I to send any fixed $\mathcal{P}$-packet along d_T edges.

Proof. First, we bound the expected number of $\mathcal{P}$-packets that participate in a T-interval. Consider any fixed edge e. Let the random variable X_t^e denote the number of packets generated at time step t that intend to cross e. Clearly, $\mathrm{E}[X_t^e] \leq \lambda$. Furthermore, let the binary random variable $X_{t,k}^g$ be 1 if and only if generator g generates at time step t a packet that has to cross e as kth edge. Then it holds that $X_t^e = \sum_{g,k} X_{t,k}^g$. Now, let us consider some fixed T-interval I that starts at time t_0. Since we require each packet to traverse $i \cdot d_T$ edges till the end of the ith T-interval in which it participates, the expected number of $\mathcal{P}$-packets that intend to cross e in I is given by

$$\begin{aligned}
\sum_g \sum_{i\geq 0} \sum_{j=0}^{T-1} \sum_{k=0}^{d_T-1} \mathrm{E}[X_{t_0-(i+1)T+j,i\cdot d_T+k}^g] &= \sum_g \sum_{i\geq 0} \sum_{k=0}^{d_T-1} T \cdot \mathrm{E}[X_{t_0,i\cdot d_T+k}^g] \\
&= \sum_g \sum_d T \cdot \mathrm{E}[X_{t_0,d}^g] \\
&= T \cdot \mathrm{E}[X_{t_0}^e] \\
&\leq T \cdot \lambda \,,
\end{aligned}$$

because $\Pr[X_{t_1,d}^g = 1] = \Pr[X_{t_2,d}^g = 1]$ for any t_1, t_2, since the injection of packets is independent of the time step.

Now we bound T so that with probability at least $1 - N^{-\phi}$ the $\mathcal{P}$-packets participating in some T-interval I are successful in I. Let us assume that $\lambda = (1-\epsilon)/\gamma$ for some $\epsilon > 0$ (resp. $\lambda = (1-\epsilon)/\log N$ in case (3)). We consider the three cases for the choice of γ and δ given in Lemma 4.3 separately.

Case 1: $\gamma = \Theta(1)$ and $\delta = \Theta(\log^\beta N)$ for some constant $0 \leq \beta \leq 1$.
From above we know that the expected number of $\mathcal{P}$-packets participating in a T-interval that cross some fixed edge is at most λT.

Suppose that $\epsilon \leq 7/8$. Since the injection of a packet is independent of the injection of other packets, we can use a Chernoff bound to show that the probability that the congestion caused by $\mathcal{P}$-packets exceeds $(1+\epsilon)\lambda T$ at some fixed edge is at most $\mathrm{e}^{-\epsilon^2\lambda T/3}$. Hence, for every $\phi > 0$ there is a $t(\epsilon,\phi) = \Theta(\frac{\phi}{\epsilon^2}\log N)$ such that for all $T \geq t(\epsilon,\phi)$, the probability is at least $1 - N^{-\phi}/2$ that the congestion caused by $\mathcal{P}$-packets in I is at most $(1+\epsilon)\lambda T$. Assume in the following that $T \geq t(\epsilon,\phi)$. Set $c_T = (1+\epsilon)\lambda T$ and $d_T = \log^\psi N$ (ψ will be specified later). According to the assumptions of Lemma 4.3, the runtime of $\mathcal{P}$, given a path collection with congestion c_T and dilation d_T, is at most $\gamma c_T + \delta d_T + O(\log^\alpha N)$ with probability at least $1 - N^{-\phi}/2$. Hence, the time required by any fixed $\mathcal{P}$-packet participating in I to be successful is at most

$$\gamma \cdot (1+\epsilon)\lambda T + \delta \log^\psi N + O(\log^\alpha N) \quad = \quad (1-\epsilon^2)T + \delta \log^\psi N + O(\log^\alpha N)$$

with probability at least $1 - N^{-\phi}$. This time bound is at most $T(1-\epsilon^2/2)$ if $T \geq \frac{2}{\epsilon^2}(\delta \log^\psi N + O(\log^\alpha N))$. (Recall that the remaining $T \cdot \epsilon^2/2$ time steps are reserved for the failed packets.)

Now suppose that $\epsilon > 7/8$. In this case, $\lambda < 1/(8\gamma)$. Since γ is assumed to be a constant, according to the Chernoff bounds the probability that the congestion caused by $\mathcal{P}$-packets exceeds $T/(4\gamma)$ can be made as small as $N^{-\phi}$ for any constant $\phi > 0$ if $T = \Theta(\phi \log N)$ is chosen large enough. Set $c_T = T/(4\gamma)$ and $d_T = \log^{\psi} N$. Analogous to above, the routing time required by the $\mathcal{P}$-packets participating in I to be successful can be made as small as $T(1-\epsilon^2/2)$ with probability of at least $1 - N^{-\phi}$ if $T = \Theta(\delta \log^{\psi} N + \log^{\alpha} N)$ is chosen large enough.

In both cases for ϵ, the bound for T is (asymptotically) minimal for $\beta + \psi = \alpha$. So altogether $T = \Theta(\frac{1}{\epsilon^2}(\phi \log N + \log^{\alpha} N))$ suffices to guarantee that any fixed $\mathcal{P}$-packet successfully manages a T-interval with probability at least $1 - N^{-\phi}$.

Case 2: $\gamma = \Theta(1)$ and $\delta = C^{\beta}$ for some constant $0 < \beta < 1$.
If $\epsilon \le 15/16$, we set $c_T = (1+\epsilon)\lambda T$ and $d_T = \log^{\psi} N$. As in case 1, the probability is at most $N^{-\phi}/2$ that the congestion in I exceeds c_T if $T = \Theta(\frac{\phi}{\epsilon^2} \log N)$ is sufficiently large. In this case, the time required by any fixed $\mathcal{P}$-packet participating in I to be successful is at most

$$\gamma \cdot (1+\epsilon)\lambda T + \delta \log^{\psi} N + \log^{\alpha} N \le (1-\epsilon^2)T + T^{\beta} \log^{\psi} N + \log^{\alpha} N$$

with probability at least $1 - N^{-\phi}$. It holds that

$$\log^{\alpha} N \le \frac{\epsilon^2}{4} \cdot T \quad \text{if} \quad T \ge \frac{4}{\epsilon^2} \cdot \log^{\alpha} N$$

and

$$T^{\beta} \log^{\psi} N \le \frac{\epsilon^2}{4} \cdot T \quad \text{if} \quad \psi = \alpha(1-\beta) \text{ and } T \ge (\epsilon/2)^{-2/(1-\beta)} \log^{\alpha} N \; .$$

Thus, if $T = \Theta((\epsilon/2)^{-2/(1-\beta)} \log^{\alpha} N)$ is sufficiently large, then

$$(1-\epsilon^2)T + T^{\beta} \log^{\psi} N + \log^{\alpha} N \le (1-\epsilon^2/2)T \; .$$

Hence, $T = \Theta((\epsilon/2)^{-2/(1-\beta)}(\phi \log N + \log^{\alpha} N))$ suffices to ensure that any fixed $\mathcal{P}$-packet successfully manages a T-interval with probability at least $1 - N^{-\phi}$.

If $\epsilon > 15/16$, we set $c_T = T/(8\gamma)$ and $d_T = \log^{\psi} N$. Similar to case 1, the probability is at most $N^{-\phi}/2$ that the congestion in I exceeds c_T if $T = \Theta(\phi \log N)$ is sufficiently large. In this case, the time required by any fixed $\mathcal{P}$-packet participating in I to be successful is at most

$$\gamma \cdot T/(8\gamma) + \delta \log^{\psi} N + \log^{\alpha} N \le T/8 + T^{\beta} \log^{\psi} N + \log^{\alpha} N$$

with probability at least $1 - N^{-\phi}$. This is at most $(1-\epsilon^2/2)T$ if

$$T^{\beta} \log^{\psi} N \le T/4 \quad \text{and} \quad \log^{\alpha} N \le T/8 \; ,$$

which is true if $\psi = \alpha(1-\beta)$ and $T = \Theta((\epsilon/2)^{-2/(1-\beta)} \log^{\alpha} N)$ is sufficiently large. Hence, if $T = \Theta((\epsilon/2)^{-2/(1-\beta)}(\phi \log N + \log^{\alpha} N))$, then also in this case any fixed $\mathcal{P}$-packet successfully manages a T-interval with probability at least $1 - N^{-\phi}$.

Case 3: the runtime of $\mathcal{P}$ is at most $C \cdot D$.
Assume that $\lambda = \frac{1-\epsilon}{\log N}$. If $\epsilon \le 7/8$, we set $c_T = (1+\epsilon)\lambda T$ and $d_T = \log N$. As above, it can be shown that the probability is at most $N^{-\phi}$ that the congestion in I exceeds c_T if $T = \Theta(\frac{\phi}{\epsilon^2} \log^2 N)$ is sufficiently large. In this case, the time required by any fixed $\mathcal{P}$-packet participating in I to be successful is at most

$$(1+\epsilon)\lambda T \cdot \log N = (1-\epsilon^2)T$$

with probability at least $1 - N^{-\phi}$.

If $\epsilon > 7/8$, then we set $c_T = T/(2\log N)$ and $d_T = \log N$. If $T = \Theta(\phi \log^2 N)$ is sufficiently large, the probability is at most $N^{-\phi}$ that the congestion in I exceeds c_T. In this case, the time required by any fixed $\mathcal{P}$-packet participating in I to be successful is at most

$$\frac{T}{2\log N} \cdot \log N \leq (1-\epsilon^2/2)T$$

with probability at least $1 - N^{-\phi}$. □

It follows for case 1 of Lemma 4.3 that if a packet following a path of length d successfully manages all T-intervals, its runtime is bounded by

$$O\left(\left(\frac{d}{\log^{\psi} N}+1\right)\frac{1}{\epsilon^2}\log^{\alpha} N\right) = O\left(\frac{1}{\epsilon^2}\left(\delta d + \log^{\alpha} N\right)\right) .$$

Similarly, for case 2, the runtime of a packet following a path of length d that successfully manages all T-intervals is bounded by

$$O\left(\left(\frac{d}{\log^{\psi} N}+1\right)(\epsilon/2)^{-\frac{2}{1-\beta}}\log^{\alpha} N\right) = O\left((\epsilon/2)^{-\frac{2}{1-\beta}}\left(d\log^{\alpha\beta} N + \log^{\alpha} N\right)\right) ,$$

and for case 3 the runtime of the packet is bounded by

$$O\left(\left(\frac{d}{\log N}+1\right)\frac{1}{\epsilon^2}\log^{2} N\right) = O\left(\frac{1}{\epsilon^2}\left(d\log N + \log^{2} N\right)\right) .$$

These time bounds match the time bounds given in Theorem 4.1 if a packet never fails. According to Lemma 4.3, the probability that a packet fails in some T-interval can be made polynomially small in N if λ is below $1/\gamma$ (resp. below $1/\log N$ in case (3)). Hence, in this case all time bounds hold w.h.p. It remains to bound the runtimes of failed packets to be able to compute the expected time a packet needs to reach its destination.

Bounding the routing time of failed packets

Next we bound the runtime of failed packets under the assumption that the injection rate is sufficiently small, that is, each individual packet is successful in every T-interval, w.h.p. For this, we will show how to apply Lemma 4.2 to bound the expected routing time of a failed packet by N^c for some constant c. Combined with the upper bound of $N^{-\phi}$ on the probability that a packet fails, this will result in a total expected routing time not much larger than the expected time that the packet requires if it does not fail in any of its T-intervals, provided that $\phi \geq c$. This yields the bounds for the expected routing time in Theorem 4.1 and implies that $\mathcal{P}'$ is stable.

Now we show how to apply Lemma 4.2. First, we change the situation that only every $\frac{2}{\epsilon^2}$th step can be used by a failed packet to a situation in which every time step can be used by a failed packet. For this, we replace each generator g by $b = 2/\epsilon^2$ generators $g_1, \ldots, g_b$, where g_i is responsible for the simulation of the behavior of g at time steps t with $(t \bmod b) + 1 = i$. Furthermore, we assume each time step to represent now b time steps in the original situation. Let us consider some fixed edge e. For any generator g that injects a packet P at time step t in the new situation, let the binary random variable Y_t^g be 1 if and only if P intends to cross e and fails in some T-interval of the original situation. From Lemma 4.3 we know that $\Pr[Y_t^g = 1]$ can be made as small as $N^{-\phi}$ for any constant $\phi > 0$.

If the probabilities $\Pr[Y_t^g = 1]$ were independent for different Y_t^g and $\Pr[Y_t^g = 1] \leq p_f$ for all t and g, we could model the injection of failed packets as a simple stochastic injection model with injection rate $\lambda = p_f \cdot b$. For $p_f \cdot b \leq 1/D(\mathcal{S})$, it would follow directly from case (2) of Lemma 4.2 (choose $K = 0$ and $\epsilon = 1 - 1/D(\mathcal{S})$) that the routing time of a failed packet is less than N^ϕ if ϕ is sufficiently large.

Coping with the dependencies

Unfortunately, there can be high dependencies among failures of packets. In order to incorporate these dependencies in the model of Lemma 4.2, we construct a dependency graph $G = (V, E)$ that has a node (g, t) for each random variable Y_t^g, and in which two nodes (g, t) and (g', t') are connected in G if and only if $t - 2T \cdot (D(\mathcal{S}) + 1)/b \leq t' \leq t + 2T \cdot (D(\mathcal{S}) + 1)/b$. Since there are $N \cdot b$ generators, the maximum degree of G is at most $(4T \cdot (D(\mathcal{S}) + 1) + b) \cdot N$, which is polynomial in N. In order to apply Lemma 4.2, we need to show that for any independent set $S \subseteq V$, the probability that $Y_t^g = 1$ for all $(g, t) \in S$ is at most $N^{-\phi|S|}$ for any constant $\phi > 0$ (depending on T).

Recall that the proof of Lemma 4.3 bounds the failure probability of a packet within a T-interval I solely by considering the injection events of packets that could have still been successful at the beginning of I and the behavior of $\mathcal{P}$ within I. Further recall that the proof of Lemma 4.3 only uses the congestion (which is upper bounded by the injection events) in I to obtain a probability bound for the success of $\mathcal{P}$. Hence, the space $\Omega_{g,t}$ of relevant outcomes that need to be considered to obtain a probability of at most $N^{-\phi}$ for the random variable Y_t^g to be 1 can be limited to contain solely injection events of packets that can participate in $\mathcal{P}$ in some T-interval together with the packet generated by g at step t. Since a packet can be alive without a failure for at most $T \cdot (D(\mathcal{S})+1)/b$ time steps, the outcome spaces of any two random variables Y_t^g and $Y_{t'}^{g'}$ with either $t' < t - 2T \cdot (D(\mathcal{S}) + 1)/b$ or $t' > t + 2T \cdot (D(\mathcal{S}) + 1)/b$ must be disjoint (that is, they do not contain a common injection event (g'', t'')). Thus, using the proof of Lemma 4.3, the probability that both of these random variables are 1 can be shown to be at most $N^{-2\phi}$. The same argument extends to any set of random variables that form an independent set in G.

In order to set the remaining parameters in the model of Lemma 4.2, we set λ equal to the given injection rate and $K = 0$ (successful packets cannot interfere with failed packets). According to Lemma 4.2, for these parameters the expected routing time of any failed packet is at most N^ϕ if ϕ is large enough. This completes the proof that for a small enough λ, even when considering failures, the expected routing time of any fixed packet is within the time bounds given in Theorem 4.1.

Recovery

We show that $\mathcal{P}'$ recovers very quickly from any worst case scenario, in a sense that all packets generated a certain amount of time after the occurrence of a worst case scenario are not (or only in a negligible way) influenced by it any more. Clearly, the shortest-in-system (SIS) rule instantly recovers from any worst case scenario because younger packets are always preferred. Similarly, we show for $\mathcal{P}'$ that after a certain amount of time, any configuration of the network has *no* influence on the runtime of the newly generated packets any more, concerning our probability bounds.

Consider any worst case scenario for the injection of packets that ends at time step t_0. Since each packet has to traverse d_T edges in each T-interval in order to remain successful, $R =$

$(\lceil \frac{D(\mathcal{S})}{d_T} \rceil + 1) \cdot T$ time steps after t_0 there can be no successful packet any more that was generated at time step t_0 or earlier. This implies that afterwards the congestion in a T-interval is only based on packets injected after the worst case scenario, which according to our stochastic injection model is independent of whatever happened during or before a worst case scenario. Since in the proof of Lemma 4.3 we used the worst case assumption that all packets have been successful so far to upper bound the congestion in a T-interval, all probability bounds in Lemma 4.3 are again valid R steps after the worst case scenario. That is, the probability of a packet to fail in a T-interval is again polynomially small in N. This ensures that sufficiently few packets fail. Once a packet fails, its age is determined by its injection time. Since SIS is used for the failed packets, a failed packet generated at a time step $t > t_0$ cannot be blocked by the packets generated during the worst case scenario or before. Hence, $\mathcal{P}'$ recovers after $(\lceil \frac{D(\mathcal{S})}{d_T} \rceil + 1) \cdot T$ time steps. Substituting the right T for each of the three cases (see Lemma 4.3) yields the recovery time given in Theorem 4.1.

4.2.3 Analysis of $\mathcal{P}'$ for Pure $\mathcal{P}$

The analysis for the successful packets is the same as above. However, since we do not have reserved time slots for the failed packets, we need a different analysis for them if $\mathcal{P}$ is pure.

Bounding the routing time of failed packets

Suppose that the injection rate is sufficiently small, that is, each individual packet is successful in every T-interval, w.h.p. For this, we will show how to apply Lemma 4.2 to bound the routing time of a failed packet by N^c for some constant c. Combined with the bound of $N^{-\phi}$ on the probability that a packet fails, this will result in an expected routing time not much larger than the time that the packet requires if it does not fail in any of its T-intervals, provided that $\phi \geq c$. This would yield the bounds for the expected routing time in Theorem 4.1 and would imply that $\mathcal{P}'$ is stable.

Now we show how to apply Lemma 4.2. Let us consider some fixed edge e. For any generator g that injects a packet P at time step t, let the binary random variable Y_t^g be 1 if and only if P intends to cross e and fails in some T-interval. From Lemma 4.3 we know that $\Pr[Y_t^g = 1]$ can be made as small as $N^{-\phi}$ for any constant ϕ. Furthermore, similar to the case that $\mathcal{P}$ is non-pure, the event $Y_t^g = 1$ causes at most $(4T \cdot (D(\mathcal{S}) + 1) + 1) \cdot N$ other random variables $Y_{t'}^{g'}$ to have a probability of $\Pr[Y_{t'}^{g'} = 1] > N^{-\phi}$, namely those with $t - 2T \cdot (D(\mathcal{S}) + 1) \leq t' \leq t + 2T \cdot (D(\mathcal{S}) + 1)$. In order to incorporate these dependencies in the model of Lemma 4.2, we again construct a dependency graph $G = (V, E)$ that has a node (g, t) for each random variable Y_t^g, and in which two nodes (g, t) and (g', t') are connected in G if and only if $t - 2T \cdot (D(\mathcal{S}) + 1) \leq t' \leq t + 2T \cdot (D(\mathcal{S}) + 1)$. Thus, the maximum degree of G is $(4T \cdot (D(\mathcal{S}) + 1) + 1) \cdot N$. To apply the model of Lemma 4.2, we set λ equal to the given injection rate and $K = (D(\mathcal{S}) + 1)T$ (successful packets live for at most $(D(\mathcal{S}) + 1)T$ time steps). According to Lemma 4.2, in this case the expected routing time of any failed packet is at most N^{ϕ} if ϕ is large enough. This completes the proof that, also for a pure $\mathcal{P}$, for a small enough λ the expected routing time of any fixed packet is within the time bounds given in Theorem 4.1.

Stability for any $\lambda < 1$

Next we consider the case that λ is arbitrarily close to 1. In this situation $\mathcal{P}$ might not be good enough to ensure that a packet manages its T-interval with high probability. However, from Lemma 4.2 (set $K = (D(\mathcal{S}) + 1) \cdot T$) it directly follows that nevertheless $\mathcal{P}'$ remains stable for any $\lambda < 1$, although the runtime of a packet might become exponential in the length of its path.

4.3 Adapting the Results to the Adversarial Model

Consider any bounded adversary of rate (w, λ). Set $\epsilon = 1 - \lambda$. For each injected packet p we choose uniformly and independently at random an initial delay of δ_p from the set $\{0, \ldots K-1\}$, where $K = (k \cdot D + w)/\epsilon^2$ and D is the length of a longest possible path. k will be specified later as it depends on the switching protocol for which we want to adapt the results. After waiting δ_p time steps in its injection buffer, p chooses the actual time step as its *new injection time* and afterwards participates in the black box simulation as described for the stochastic injection model. All that has to be shown is that if k is chosen sufficiently large, then the congestion within a T-interval is sufficiently small, so that both the packets participating in the static protocol and the failed packets can be routed efficiently. This will be done by the following lemma.

Lemma 4.4 *If $k \geq T(1/d_T + 1/D)$ then the expected congestion within any fixed T-interval is at most $\lambda' T$, where $\lambda' \leq 1 - \epsilon/2$.*

Proof. Consider some fixed edge e and T-interval I, and let t_0 be the first step of I. A packet is said to *participate* in I if it participates in the static protocol in I and it intends to cross e in I. From the construction of the black box transformation it follows that a packet injected at time t that has e as the ith edge on its routing path participates in I if and only if $\lceil \frac{t}{T} \rceil \cdot T + \frac{i}{d_T} \cdot T \in I$. Thus, only packets with new injection times in the interval $[t_0 - (T + (D/d_T) \cdot T), t_0]$ can participate in I. Since the new injection time can be at most $K - 1$ time steps ahead of the original injection time, we conclude that only packets with original injection times in the interval $[t_0 - (D/d_T + 1)T - (K-1), t_0]$ can participate in I. Suppose that $k \geq T(1/d_T + 1/D)$. According to the adversarial injection model, at most

$$
\begin{aligned}
(1-\epsilon)w \cdot \left\lceil \frac{(D/d_T+1)T + K}{w} \right\rceil &= (1-\epsilon)w \cdot \left\lceil \frac{(D/d_T+1)T + (k \cdot D + w)/\epsilon^2}{w} \right\rceil \\
&\leq (1-\epsilon)w \cdot \left\lceil \frac{(1 + 1/\epsilon^2)k \cdot D + w/\epsilon^2}{w} \right\rceil \\
&\leq (1-\epsilon) \cdot (k \cdot D + w) \cdot (1 + 1/\epsilon^2) \\
&= (1-\epsilon)(1+\epsilon^2) \cdot K \leq (1 - \epsilon/2) \cdot K
\end{aligned}
$$

packets can be injected that intend to cross e in I. Since each of these packets chooses a random initial delay out of a range of size K and there are at most T time steps for the new injection time of any given packet such that it participates in I, the expected number of packets participating in I is at most

$$\frac{T}{K} \cdot (1 - \epsilon/2) \cdot K = (1 - \epsilon/2)T \; .$$

□

Since the packets choose their initial delays independently at random, the lemma can be used to prove also bounds on the congestion in any T-interval that hold with high probability. In this case, the same analysis as in Section 4.2 can be used to show that, for instance, for $\gamma = \Theta(1)$ and $\delta = \Theta(\log^\beta N)$ the delay of any packet is bounded by $O(\frac{1}{\epsilon^2}(\frac{1}{\epsilon^2}\delta D + w + \log^\alpha N))$, w.h.p., if $\lambda \leq (1-\epsilon)/\gamma$. As a consequence, the results on the stability of these protocols hold also in the adversarial model. However, note that the runtime of a packet now depends on D rather than the length of its individual routing path.

4.4 Conclusions

In this chapter, we presented a black box transformation scheme applicable to every static online switching algorithm. Our results show that it is important for static switching protocols to know the exact constant in front of the C in the runtime bound, since this determines up to which injection rate the resulting dynamic protocol ensures a small delay for the packets. It is an interesting question how large this constant is for the known static switching protocols, and whether there exist static switching protocols with runtime $C + O(D + \log N)$.

A drawback of our black box transformation is that if the number of packets injected in a certain time interval exceeds some limit, many packets will suddenly have a high delay, since they fail and therefore SIS is applied to them. Are there transformations that ensure a much smoother behavior for the delays? Answers to this question will be partly given in the next chapter.

Chapter 5

Locally Efficient Switching Protocols

Most of the work done in the area of static switching concentrates on minimizing the runtime of the whole schedule rather than minimizing the runtime of individual packets, using global parameters such as the congestion and dilation of a path collection. In this chapter, we study the problem of minimizing the routing time of individual packets, using local parameters. In fact, we present the first close to optimal, truly online switching protocols for the following problem:

Assume that a fixed collection of paths is given. For every path p in this collection, let c_p denote the maximum, over all edges e in p, of the number of paths crossing e, and let d_p denote the length of p. Find a schedule (that does *not* require to know c_p and d_p) such that the routing time of a packet following a path p merely depends on c_p and d_p.

We also consider a dynamic variant of this problem together with an injection model that covers all stochastic and adversarial injection models known to us.

5.1 Introduction

In almost all papers on the subject of static packet switching, the performance of a protocol is expressed in terms of global parameters such as the *congestion* (the maximum number of paths that cross an edge) and *dilation* (the length of the longest path used by a packet). However, in many situations it would be much more desirable to be able to send packets along their paths with an end-to-end delay that only depends on the communication traffic along their own path, and not on the situation in the whole network. For instance, it would be certainly very bad if communication in the Internet or a parallel computer depended on the most congested part in it.

Locally efficient protocols could be used, for instance, to exploit topological locality in parallel systems in the sense that processors that are close together, according to the topology of the network, can exchange data much faster than processors that are far apart. Topological locality can be due to a communication sensitive mapping of processes (*task mapping*) or global variables (*data management*) to processors. Several papers have already dealt with the question of how to exploit locality for data management in distributed systems [ABF93, ABF96, LRWY96, MMVW97]. However, none of them considers the problem of how to actually send messages through the system. They only deal with minimizing the distance between a datum and its requesting process, or how to select paths to minimize the congestion at the links caused by the requests. In order to exploit topological locality, it is important to have a routing algorithm that is able to prefer packets accordingly.

5.1.1 Models and Problems

In this chapter, we again consider arbitrary network topologies modeled as undirected graphs with n nodes. The nodes represent switches, and the edges represent bidirectional communication links with buffers for outgoing packets on either side. As above, the routing is performed in a synchronous "store and forward" manner. In every step, each edge can be crossed by at most one packet in each direction. Since a packet has to store its destination, it has to consist of at least $\log n$ bits. Thus, we assume in the following that all packets are of size $O(\log n)$, i.e., $O(\log n)$ bits can cross an edge in one time step. (This will be important for our algorithms, since apart from the packets we will also send special control packets that can carry a certain amount of information.)

We will present switching protocols in which the nodes locally decide which packets to move forward in each step, i.e., a decision only depends on the routing information carried by the packets and on the local history of the execution. Recall that these algorithms are called *online* algorithms.

The injection model

In a dynamic setting, paths are continuously injected into the system. An injection model is called *bounded* if there are $T, B \in \mathbb{N}$, such that for any time interval I of length T and edge e, the number of paths generated during I that cross e is at most B (in each direction), either w.h.p (which means "with probability at least $1 - n^{-\alpha}$", where $\alpha > 0$ is an arbitrary constant) in case of stochastic injections, or with certainty in case of adversarial injections. The *injection rate* λ of the system is defined as B/T. A switching protocol is called *stable* if the (expected) number of packets stored at the edges does not increase without bound in the course of time. Clearly, a necessary condition for a protocol to be stable in our model is that $\lambda \leq 1$.

The packet switching model

We use the following two models:

Static packet switching: Assume that a fixed collection of paths is given. For every path p in this collection, let c_p denote the maximum, over all edges e in p, of the number of paths crossing e, and let d_p denote the length of p.

Dynamic packet switching: Assume that paths are continuously injected into the system according to an arbitrary injection model. Along each path, one packet has to be sent. Once a packet traversed its path, this path is removed from the system. For each path p, we define c_p as the maximum number of paths crossing an edge of p that were in the system when p was injected. d_p denotes the length of p.

Obviously, for both models a protocol is worst case optimal if it guarantees a runtime of $O(c_p + d_p)$ for every packet following a path p. (However, this is not an absolute lower bound for every individual packet since we can always give a high priority to a specific packet such that its end-to-end delay is d_p.) Often it is assumed that, in addition to local information, online algorithms know the size, congestion and dilation of a switching problem. We are more strict here, because we assume that nothing about the switching problem is known in advance. (The nodes only have to know the size of the network, which is independent of the switching problem.) This assumption is important, since in general it might be very difficult, if not impossible for the processors to have sufficient knowledge about the congestion at other parts of the system. For instance, this might be very difficult if many parallel programs are executed simultaneously by the same distributed system. Also, processes and shared objects might be moved while the application is running to improve, for instance, the balance of the work load of the processors.

5.1.2 Previous Results

In a pioneering paper, Leighton, Maggs, and Rao [LMR88] showed that there is an offline schedule for any simple path collection with dilation D and congestion C that sends all packets along their paths (one packet per path) in time $O(C + D)$. Since then, many online switching protocols have been presented for various classes of path collections (see, e.g., [LMRR94, MV95, CMSV96, RT96, OR97]). To give just two examples, in [CMSV96] Cypher et al. present an online, buffer-less algorithm that works for any simple path collection of size n. Their algorithm requires $O\left(D \cdot \log\log n + C + \frac{\log n \cdot \log\log n}{\log\log(C \cdot D)}\right)$ steps, w.h.p., to deliver all packets to their destinations, provided that the communication links have bandwidth $\Theta\left(\log(C \cdot D)/\log\log(C \cdot D)\right)$. In [OR97] Ostrovsky and Rabani present an online protocol with runtime $O(C + D + \log^{1+\epsilon} n)$, w.h.p., for any constant $\epsilon > 0$.

If the local congestion and dilation of the paths is known in advance for every path, the protocols presented in the papers cited above can be easily transformed into locally efficient online protocols. That is, we can replace C and D by the local parameters c_p and d_p in the runtime bound for each packet P. (Simply partition the packets into sets with $\max[c_p, d_p] \in [2^i, 2^{i+1} - 1]$, and route these sets one after the other.) However, if the local congestion and dilation of the paths is *not* known in advance, it was not known how to transform the existing protocols into locally efficient switching protocols.

Theoretical results about dynamic switching protocols for interconnection networks are relatively new. Several papers have already dealt with dynamic switching protocols for butterflies, meshes, and hypercubes (see, e.g., [ST91, Mit94, KL95, BU96]), and so-called Markovian networks [ST91, HW95]. The first analysis of a protocol for arbitrary networks can be found in [SV96]. In [SV96], a protocol is presented that ensures for a stochastic injection model that only allows the injection of shortest paths that, in the steady state, a packet that has to travel a distance of d is delivered in time $O(d)$, w.h.p. In [BFU96] Broder et al. present conditions that are sufficient for the stability of dynamic packet switching algorithms. Their approach reduces the problem of steady state analysis to the easier question of static switching.

Whereas the results above are based on a stochastic injection model, there have also been some results based on a model called *adversarial queuing theory*. In this model, it is possible to have an adversary that can select the paths of the packets within a certain limit. This approach was introduced by Borodin et al. in [BKR$^+$96]. Further results have been presented by Andrews et al. [AAF$^+$96]. Among other things, they show that there are simple greedy protocols that are stable for every network, but other commonly-used protocols (such as FIFO) are not stable for every network.

5.1.3 New Results

We present a protocol called the *FLP (first local packet switching) protocol* that achieves the following result:

Theorem 5.1 *For any simple path collection, the FLP protocol routes any fixed packet along its path p in expected time $O(c_p + d_p \cdot log(c_p + d_p))$. Furthermore, an additive delay of t to this runtime only occurs with a probability of at most $2^{-\Theta(\sqrt[3]{t/\log t})}$.*

The main idea behind this protocol is a strategy that allows the packets to guess correctly with high probability whether they should be given preference or not. The main problem in the analysis of the protocol is to get rid of certain correlations among the packets. For this, we additionally send special *ghost packets*. We note that the FLP protocol is the only protocol presented in this chapter that may violate the restriction on the number of bits for the packets (some control packets may need more than $O(\log n)$ bits if c_p or d_p is in the order of $n^{\omega(1)}$).

Based on the FLP protocol, we present a protocol called the *SLP (static local packet switching) protocol* that achieves a faster runtime than the first one in case that c_p or d_p is larger than $\log n$.

Theorem 5.2 *For any simple path collection, the SLP protocol routes any fixed packet along its path p in expected time $O(c_p + d_p \cdot \min[\log(c_p + d_p), \log\log n])$. Furthermore, an additive delay of t to this runtime only occurs with a probability of at most $2^{-\Theta(\sqrt[3]{t/\log t})}$ if $\sqrt[3]{t/\log t} \le \log n$, and $n^{-\Theta(\log t - \log\log n)}$ otherwise.*

Apart from static protocols, we also present a dynamic protocol for shortcut-free paths, called *DLP (dynamic local packet switching) protocol*. A path collection is called *short-cut free* if no part of a path can be used as a short-cut of a part of another path. (This holds, for instance, if no two paths meet more than once.)

Theorem 5.3 *For any injection model restricted to short-cut free paths, the DLP protocol routes any fixed packet along its path p in expected time $O(c_p + d_p \cdot \log\log n)$. Furthermore, an additive delay of t to this runtime only occurs with a probability of at most $2^{-\Theta(\sqrt{t/\log t})}$ if $\sqrt{t/\log t} \le \log n$, and $n^{-\Theta(\log t - \log\log n)}$ otherwise. Moreover, the protocol is stable up to a constant injection rate for any bounded injection model.*

The advantage of our algorithm over previously presented algorithms is that it does not only predict results for the steady state (as usually done under stochastic injection models) or the worst case (as done under adversarial models). It can give the runtime of a packet in terms of *local* parameters at *any* time point of the dynamic switching process for *any* injection model.

5.2 Static Packet Switching

In this section, we prove the theorems in Section 5.1.3 for the static packet switching model.

5.2.1 The FLP Protocol

Let us consider all paths to be partitioned into sets $S_1, S_2, \ldots$, with S_i containing all paths p with $\max[c_p/i, d_p] \in [2^{i-1}, 2^i)$. Our protocol works in *rounds*, starting with round 1. For every $r \ge 1$, the rth round has the task to route all packets with paths in S_r to their destinations, *without* knowing which packets actually belong to S_r. We will show that

(1) round r takes $O(r \cdot 2^r)$ time steps, and

(2) any participating packet having a path in $S_{r'}$ with $r' \le r$ successfully reaches its destination with a probability of at least $1 - 2^{-\Theta(\sqrt[3]{2^r})}$.

This means that

- the expected time to route a packet with path in S_r is bounded by

$$\sum_{i=1}^{r} O(i \cdot 2^i) + \sum_{i=r+1}^{\infty} O(i \cdot 2^i) \cdot 2^{-\Theta(\sqrt[3]{2^i})} = O(r \cdot 2^r) = O\left(\max\left[\frac{c_p}{r},\ d_p\right] \cdot r\right)$$
$$= O(c_p + d_p \cdot \log(c_p + d_p)) \ ,$$

and

- a deviation of an additive t from this bound only occurs with a probability of at most $2^{-\Theta(\sqrt[3]{t/\log t})}$.

Thus it remains to prove items (1) and (2) above. Let us consider a fixed round $r \geq 1$. Let $T = \sqrt[3]{2^r}$. Every edge is assumed to have a bandwidth b of $\beta \log T$, i.e., every edge can forward b packets at every time step. (Obviously, each of these time steps can be simulated in $\beta \log T$ time steps by edges with bandwidth 1.) Apart from the usual packets (called *real packets* in the following), so-called *ghost packets* are generated at each node. In fact, each node generates $T^3 \cdot \log T$ ghost packets for each of its links. Each ghost packet is assigned a delay $\delta \in [5T^3]$, chosen independently and uniformly at random (i.u.r.). Ghost packets, however, never really leave a node. As we will see later, their only purpose is to keep down certain correlations among the real packets. Furthermore, each real packet P is followed by a so-called *counting packet* P'. The packet P' counts the time steps in which at least b packets want to cross an edge of the path of P (this information has to be collected by the nodes of the network). Let us now describe how round r works. In the following, $[x]$ for some integer $x \in \mathbb{N}$ denotes the set $\{1, \ldots, x\}$, and $[x, y]$ for some integers $x < y$ denotes the set $\{x, \ldots, y\}$.

Round r of the FLP protocol:

- **Phase 1:** Every packet that has not reached its destination yet, chooses i.u.r. an initial delay $\delta \in [4T^3]$ with $T = \sqrt[3]{2^r}$. A real packet that is assigned a delay of δ waits in its current edge buffer for δ time steps and then tries to move on without waiting until it reaches its destination or traversed T^3 edges.

  If, together with the ghost packets, more than b packets want to cross an edge e at the same time, then all of the involved real packets will remain in the outgoing buffer of e until the end of the phase. Such packets are called *unsuccessful*.

- **Phase 2:** For every real packet P, we send a counting packet P' along P's path. In this phase, we do not use ghost packets any more. P' uses the same initial delay as P, and then it moves on without waiting until it reaches the current edge buffer of P. Obviously, this is always possible for P', as Phase 2 is a playback of Phase 1.

  If a counting packet P' counts more than $(\beta T)^2$ events, where at least b packets (*not* counting P) try to cross one of the edges along the path of the corresponding real packet P at the same time, and P was unsuccessful in the first phase, P will be *dumped*, which means that it stays in its current edge buffer for the rest of the round.

- **Phase 3:** The third phase consists of $2T$ *passes*. For every pass, every unsuccessful but not dumped real packet chooses i.u.r. an initial delay from a range of $[2(\beta T)^2]$. Afterwards, it attempts to move along the next T^2 edges on its path. Once a packet reaches its destination or traversed T^3 edges in the whole round, it stops and is declared *successful*.

In order to show that the above strategy fulfills the requirements stated in (1) and (2) above, we prove several lemmas. As a worst case assumption, we assume that the local congestion of a path p is still c_p during every round, i.e., every path that shares an edge with p is still participating in that round. Furthermore, we assume that T is beyond some sufficiently large constant. Since below this constant all rounds take constant time, this does not affect our runtime bounds asymptotically.

We frequently use the following notation. Given a packet P, a subpath $q = e_1 \dots e_\ell$ of P's path from edge e_1 to e_ℓ is called its *active segment* if q represents the path which P attempts to traverse during this round. A *site* is an ordered pair (e, t) with e being an edge and t being a time step. A packet *aims for* a site (e, t) if it selects a random delay so that it intends to cross e at time t. Furthermore, a packet is said to *participate* in a site (e, t) if it indeed crosses it at time t. Define a *run* of P to be the set of sites which P will aim for for some fixed delay δ, i.e., a run consists of sites $(e_1, \delta), (e_2, \delta + 1), \dots, (e_\ell, \delta + \ell - 1)$. A site is called *$d$-site* (resp. *$>d$-site, $\geq d$-site*) if exactly d (resp. more than d or at least d) packets participate in that site.

In the following, we will prove items (1) and (2) above for some fixed round r.

Lemma 5.4 *Consider any fixed $r' \leq r$. A packet with a path in $S_{r'}$ is dumped with probability at most e^{-cT} for any constant c depending on b.*

Proof. First, we give a high-level description of the proof. The counting packet of a packet P counts the number of $\geq b$-sites in order to decide if P gets dumped or not. Thus, we have to bound the number of $\geq b$-sites along the edges of P's active segment q. Since the length of q is at most T^3 and the packets can aim for at most $5T^3$ different sites per edge, there are at most $5T^6$ sites that have to be considered. The proof is divided into two parts.

In the first part we show that the number of $>b$-sites along q is at most T with a probability of at least $1 - \mathrm{e}^{-cT \cdot \log T}$. Because packets involved in a $> b$-site remain in the corresponding edge buffer, the $> b$-sites can be regarded as independent of each other for an upper bound on their number. Thus, the probability bound can be obtained by the use of Chernoff bounds.

In the second part we bound the number of b-sites along q. This is much more difficult, because now there can be high dependencies among the sites of q. For example, the event that a packet traverses a b-site at time t may significantly influence the event whether there is a b-site at time $t + 1$ at its next edge. In the case that the paths are not short-cut free, there may be also high dependencies among different runs. In order to cope with the dependencies, we need the ghost packets. These packets allow us to show that the probability of a packet to be involved in more than $\beta^{3/2}T$ different b-sites is at most $\mathrm{e}^{-c'T}$. This enables us to bound the number of b-sites along q by $\beta^2 T \cdot (T - 1)$ with a probability of at least $1 - \mathrm{e}^{-c'' \cdot T}$. Thus, the number of $\geq b$-sites along the edges of q can be upper bounded by $(\beta T)^2$ with a probability of at least $1 - \mathrm{e}^{-cT}$. We now give a detailed proof.

Let P be a packet with active segment q in $S_{r'}$, $r' \leq r$. First, we upper bound the number of $>b$-sites on the active segment of P. Consider marking any m sites along the edges of q. There are at most T^3 edges in q and for each of these edges there are $4T^3 + T^3 = 5T^3$ possible time steps at which real packets might arrive. Hence, there are at most $\binom{5T^6}{m}$ possibilities of marking m sites.

Consider some fixed set of m marked sites, and suppose the packets that can aim for any one of the marked sites to be numbered from 1 to k. Let the binary random variable X_i be 1 if and only if packet i aims for any one of the marked sites, and let $X = \sum_{i=1}^{k} X_i$. Since $r' \leq r$, there are at most $r \cdot 2^r = 3T^3 \log T$ real packets that cross an edge of q. Together with the $T^3 \log T$ ghost packets we therefore have a total of at most $4T^3 \log T$ packets that cross an edge of q. Since each of these packets chooses a random delay from a range of size at least $4T^3$, the expected number of packets which aim for any one site is at most $\log T$. Therefore, $\mathrm{E}[X] \leq m \log T$. Since a packet can only participate in one $>b$-site, we have: if all m marked sites are $>b$-sites, then $X > b \cdot m$. As the packets choose their delays independently at random, we can apply the Chernoff bounds to obtain

$$\Pr[X > b \cdot m] \;\leq\; \left(\frac{\mathrm{e}^{\beta-1}}{\beta^\beta}\right)^{m \log T} .$$

Let $m = T$. Then we have, for $\beta = \max[5 + c, \mathrm{e}^2]$,

$$\begin{aligned} \Pr[\geq m \text{ sites are } >b\text{-sites}] &\leq \binom{5T^6}{m}\left(\frac{\mathrm{e}^{\beta-1}}{\beta^\beta}\right)^{m \log T} \\ &\leq \left(\frac{5\mathrm{e}T^6}{m}\right)^m \cdot \mathrm{e}^{-\beta m \log T} \leq \mathrm{e}^{m(5 \ln T + \ln(5\mathrm{e})) - \beta m \log T} \\ &\leq \mathrm{e}^{(5-\beta)m \log T} \leq \mathrm{e}^{-cT \log T} \end{aligned}$$

if T is sufficiently large. Thus, the number of $>b$-sites along q can be upper bounded by T with a probability of at least $1 - \mathrm{e}^{-cT \log T}$. In order to prove Lemma 5.4 it remains to bound the number of b-sites along q (note that a packet counts the number of $\geq b$-sites in order to decide if it gets dumped or not!). This is more difficult than bounding the number of $>b$-sites, because now there may be dependencies causing positive correlations among the sites of q. In order to cope with these dependencies, we need the ghost packets. These packets allow us to show the following proposition.

Proposition 5.5 *For any constant $\gamma > 0$ there is a constant $\beta > 0$ for b such that the probability that some fixed packet participates in more than $\beta^{3/2}T$ many b-sites in Phase 1 is at most $\mathrm{e}^{-\gamma T}$.*

Proof. Consider a fixed packet P with active segment $q = e_1 \dots e_\ell$. P may belong to any $S_{r'}$ (that is, we do not demand that $q \in S_{r'}$ for some $r' \leq r$). In the following, let the random variable X_i denote the number of packets participating in the site of P's run at edge e_i *including* P, $i \in [\ell]$, and let $X = \sum_{i=1}^{\ell} X_i$. In order to bound the number of b-sites in which P participates, we prove the following claim.

Claim 5.6 *For any $i \in [\ell]$ and set $J \subseteq [\ell] \setminus \{i\}$ and any $c_j \in \mathbb{N}_0$ with $j \in J$ it holds*

$$\Pr\left[X_i > b \mid (\bigwedge_{j \in J} X_j = c_j) \;\wedge\; X_i \geq b\right] \geq \frac{1}{6\beta} \cdot \Pr\left[X_i = b \mid (\bigwedge_{j \in J} X_j = c_j) \;\wedge\; X_i \geq b\right] .$$

Proof. W.l.o.g., we assume that $J = \{1, \dots, k\}$ and $i = k + 1$ for some k. Let M be the set of all possibilities to select packets such that $X_j = c_j$ for all $j \in [k]$. Each of these possibilities may be represented by a *configuration* $C = (C_1, \dots, C_k)$, where C_j is the set containing those c_j packets that participate in site j. Let E_C be the event that configuration C is true. Then

$$\Pr[\bigwedge_{j \in [k]} X_j = c_j] = \Pr[\bigcup_{C \in M} E_C] .$$

Now, let the random variable Z_r denote the number of real packets that participate in site $k+1$, and let Z_g denote the number of ghost packets that aim for (resp. participate in) site $k+1$.

Given a fixed Z_r, the probability that exactly $b - Z_r$ ghost packets aim for site $k+1$ is either 0 if $Z_r > b$ or

$$\binom{T^3 \log T}{b - Z_r} \left(\frac{1}{5T^3}\right)^{b-Z_r} \left(1 - \frac{1}{5T^3}\right)^{T^3 \log T - (b - Z_r)} .$$

Thus, for any fixed configuration C,

$$\Pr[X_{k+1} > b \mid E_C \ \wedge \ Z_r > b \ \wedge \ X_{k+1} \geq b] \geq \Pr[X_{k+1} = b \mid E_C \ \wedge \ Z_r > b \ \wedge \ X_{k+1} \geq b] = 0$$

and otherwise

$$\begin{aligned}
&\frac{\Pr[X_{k+1} > b \mid E_C \ \wedge \ Z_r \leq b \ \wedge \ X_{k+1} \geq b]}{\Pr[X_{k+1} = b \mid E_C \ \wedge \ Z_r \leq b \ \wedge \ X_{k+1} \geq b]} \\
&= \frac{\Pr[X_{k+1} > b \mid E_C \ \wedge \ Z_r \leq b]}{\Pr[X_{k+1} = b \mid E_C \ \wedge \ Z_r \leq b]} \\
&\geq \frac{\Pr[Z_g = b + 1 - Z_r \mid E_C \ \wedge \ Z_r \leq b]}{\Pr[Z_g = b - Z_r \mid E_C \ \wedge \ Z_r \leq b]} \\
&= \frac{\sum_{i=0}^{b} \Pr[Z_r = i \mid E_C] \cdot \Pr[Z_g = b + 1 - Z_r \mid E_C \ \wedge \ Z_r = i]}{\sum_{i=0}^{b} \Pr[Z_r = i \mid E_C] \cdot \Pr[Z_g = b - Z_r \mid E_C \ \wedge \ Z_r = i]} \\
&= \frac{\sum_{i=0}^{b} \Pr[Z_r = i \mid E_C] \cdot \binom{T^3 \log T}{b+1-i} \left(\frac{1}{5T^3}\right)^{b+1-i} \left(1 - \frac{1}{5T^3}\right)^{T^3 \log T - (b+1-i)}}{\sum_{i=0}^{b} \Pr[Z_r = i \mid E_C] \cdot \binom{T^3 \log T}{b-i} \left(\frac{1}{5T^3}\right)^{b-i} \left(1 - \frac{1}{5T^3}\right)^{T^3 \log T - (b-i)}} \\
&\geq \frac{T^3 \log T - b}{b+1} \cdot \frac{1/(5T^3)}{1 - 1/(5T^3)} \geq \frac{\log T}{6b} = \frac{1}{6\beta}
\end{aligned}$$

if T is sufficiently large, since $\binom{T^3 \log T}{b+1-i} / \binom{T^3 \log T}{b-i} \geq \frac{T^3 \log T - b}{b+1}$ for all $i \in \{0, \ldots, b\}$. Therefore,

$$\begin{aligned}
&\frac{\Pr[X_{k+1} > b \mid (\bigwedge_{j \in [k]} X_j = c_j) \ \wedge \ X_{k+1} \geq b]}{\Pr[X_{k+1} = b \mid (\bigwedge_{j \in [k]} X_j = c_j) \ \wedge \ X_{k+1} \geq b]} \\
&= \frac{\sum_{C \in M} \Pr[X_{k+1} > b \mid E_C \ \wedge \ X_{k+1} \geq b]}{\sum_{C \in M} \Pr[X_{k+1} = b \mid E_C \ \wedge \ X_{k+1} \geq b]} \\
&\geq \frac{\frac{1}{6\beta} \sum_{C \in M} \Pr[E_C] \cdot \Pr[X_{k+1} = b \mid E_C]}{\sum_{C \in M} \Pr[E_C] \cdot \Pr[X_{k+1} = b \mid E_C]} \\
&\geq \frac{1}{6\beta} .
\end{aligned}$$

□

The above claim implies that for each additional $\geq b$-site in which P is involved, the probability that this is a $> b$-site (and therefore P stops) is at least a constant times the probability that this is a b-site (and therefore P can continue):

Let A_b (resp. $A_{\geq b}$, $A_{>b}$) be the event that some given site i is a b-site (resp. $\geq b$-site, $>b$-site), assuming any event $\bigwedge_{j \in J} X_j = c_j$ with $J \subseteq [\ell] \setminus \{i\}$. Since

$$\Pr[A_{>b} \mid A_{\geq b}] + \Pr[A_b \mid A_{\geq b}] = 1$$

and

$$\Pr[A_{>b} \mid A_{\geq b}] \geq \tfrac{1}{6\beta} \Pr[A_b \mid A_{\geq b}]$$

according to Claim 5.6, it follows that

$$\Pr[A_b \mid A_{\geq b}] \leq \frac{1}{1+\frac{1}{6\beta}} = 1 - \frac{1}{6\beta+1} \leq \mathrm{e}^{-\frac{1}{6\beta+1}} .$$

Set $p = \mathrm{e}^{-1/6\beta}$. Now we are ready to bound the number of b-sites along the active segment q which P will visit on its run. Note that the number of b-sites in which P participates is at least the number of $\geq b$-sites minus one, because P stops as soon as it participates in a $> b$-site. Let $D = \mathcal{P}([\ell])$ represent the set of all possible distributions of $< b$ and $\geq b$-sites along P's run in the following way: for any $U \in D$, let the event E_U denote the set of all outcomes such that U contains exactly those sites that are $\geq b$-sites. Let the binary random variable Y_i be one if and only if site i is a b-site and there is no site $j < i$ that is a $> b$-site. Furthermore, let $Y = \sum_{i=1}^{\ell} Y_i$. Then it holds:

$$\begin{aligned} \Pr[Y \geq k] &\leq \sum_{U \in D} \Pr[E_U] \cdot \Pr[\bigwedge_{\text{first } k \text{ numbers } i \in U} Y_i = 1 \mid E_U] \\ &= \sum_{U \in D} \Pr[E_U] \cdot \prod_{\text{first } k \text{ numbers } i \in U} \Pr[Y_i = 1 \mid E_U \wedge (\bigwedge_{j \in U:\, j<i} Y_j = 1)] \\ &\leq \sum_{U \in D} \Pr[E_U] \cdot p^k = p^k \end{aligned}$$

Hence, for $\beta^{3/2}/(6\beta+1) \geq \gamma$ we get

$$\Pr[Y \geq \beta^{3/2}T] \leq p^{\beta^{3/2}T} = \mathrm{e}^{-\frac{\beta^{3/2}}{6\beta+1}\cdot T} \leq \mathrm{e}^{-\gamma T} .$$

This completes the proof of Proposition 5.5. □

We are now ready to prove Lemma 5.4. First, we define a set of binary random variables $X_1, \ldots, X_m$. Each X_i corresponds to one of the at most $5T^6$ possible sites along the edges of q (i.e., $m \leq 5T^6$). X_i is equal to 1 if and only if at least $b/2$ packets *aim* for site i. Clearly,

$$\Pr[X_i = 1] \leq \binom{4T^3 \log T}{b/2} \left(\frac{1}{4T^3}\right)^{b/2} \leq \left(\frac{2\mathrm{e}}{\beta}\right)^{b/2} .$$

Let $X = \sum_{i=1}^{m} X_i$. Then $\mathrm{E}[X] \leq m \cdot (2\mathrm{e}/\beta)^{b/2} \leq 1$ if $\beta \geq 4\mathrm{e}$ and T is sufficiently large. Let the random variable X' denote the maximum number of sites along q for each of which a set of at least $b/2$ packets aiming for it can be marked that is disjoint to all other sets of marked packets. Clearly, $X' \leq X$. On the other hand, if there are at least k many b-sites and no packet participates in more than d of them, then there are at least k/d of these sites for each of which a set of $b/2$ packets can be marked that is disjoint to all other sets of marked packets. Assume that this is not the case, and let $k' < k/d$ be the maximum number of b-sites with the property above. The corresponding $k' \cdot b/2$ marked packets can be used to construct a participation by at least $b/2$ packets for at most $k' \cdot b/2 \cdot d/(b/2) = k' \cdot d < k$ sites. Thus, there must be at least one additional b-site that has at least $b/2$ packets that have not been marked by the k' selected sites, which contradicts the assumption.

Hence, to summarize our observations, if there are at least k many b-sites along q and none of the participating packets participates in more than d of them, then $X' \geq k/d$. Furthermore, the corresponding X_i's are negatively correlated, as the sets of marked packets are disjoint. Since $\mathrm{E}[X'] \leq 1$ if β and T are sufficiently large, it follows from the Chernoff bounds that

$$\Pr[X' \geq k/d] \leq \left(\frac{\mathrm{e}}{k/d}\right)^{k/d} .$$

If we set $k = \beta^2 T(T-1)$ and $d = \beta^{3/2}T$, we obtain

$$\Pr[X' \geq \beta^{1/2}(T-1)] \leq \left(\frac{\mathrm{e}}{\beta^{1/2}(T-1)}\right)^{\beta^{1/2}(T-1)} \leq \mathrm{e}^{-\gamma T}$$

for any constant $\gamma > 0$ if T and β are sufficiently large.

According to Proposition 5.5, the probability that some fixed packet participates in more than $\beta^{3/2}T$ many b-sites is at most $\mathrm{e}^{-\gamma' T}$ for any constant γ' depending on β. Since the number of packets whose active segments share an edge with q can be at most $T^3 \cdot 3T^3 \log T$, the probability that there are at least $\beta^{3/2}T \cdot \beta^{1/2}(T-1) = \beta^2 T(T-1)$ many b-sites along the edges of q can be made as small as $\mathrm{e}^{-\gamma'' T}$ for any constant $\gamma'' > 0$.

Set $c = \gamma'' - 1$ and assume T to be sufficiently large. Then, together with the bound on the number of $>b$-sites, we obtain a probability of at most e^{-cT} for any constant c (depending on β) that packet P has more than $\beta^2 T(T-1) + T \leq (\beta T)^2$ many $\geq b$-sites along its active segment. This proves Lemma 5.4. □

Lemma 5.7 *Consider any active segment q in $S_{r'}$ with $r' \leq r$. It holds for any constant c depending on b that, with probability at least $1 - \mathrm{e}^{-cT^2 \cdot \log T}$, every edge of q is contained in at most $2(\beta T)^2 \cdot \log T$ active segments of packets that were unsuccessful in Phase 1 but not dumped.*

Proof. Consider some fixed edge e of q. First, we bound the expected number of active segments of unsuccessful, non-dumped packets containing e. For this, consider some fixed packet P (with a path segment that is not required to be in $S_{r'}$ for some $r' \leq r$). Let A_P be the event that it fails without being dumped, that is, from the start of its active segment to the place where it fails the other packets form at most $(\beta T)^2$ many $\geq b$-sites. Let M denote the set of all possible assignments of the packets to the sites of P, and let E_C be the event that a given assignment $C \in M$ is true, in a sense that the packets would indeed participate in the corresponding sites without P. Then

$$\Pr[A_P] = \sum_{C \in M} \Pr[E_C] \cdot \Pr[A_P \mid E_C] \leq \sum_{C \in M} \Pr[E_C] \cdot \frac{(\beta T)^2}{4T^3} = \frac{\beta^2}{4T} ,$$

since the probability that P aims for any one of the $(\beta T)^2$ first $\geq b$-sites along the path of P given by C is at most $(\beta T)^2/(4T^3)$. As the edge e chosen above belongs to a path in $S_{r'}$ for some $r' \leq r$, e is contained in at most $3T^3 \log T$ active segments. Hence, the expected number of unsuccessful, non-dumped packets containing e in their active segment is at most $(\beta T)^2 \log T$.

Consider now any set of packets $P_1, \ldots, P_k$ crossing e. Set $p = \beta^2/(4T)$. Let M denote the set of all possible ways of assigning packets, excluding the packets $P_1, \ldots, P_k$, to the sites along the active segments of $P_1, \ldots, P_k$, and let E_C be the event that a given assignment $C \in M$ is true (in a sense that the packets would indeed participate in the corresponding sites without $P_1, \ldots, P_k$). Furthermore, let M_i denote the set of all possible assignments of the packets, excluding P_i, to the sites along the active segment of P_i. Since the packets choose their delays independently at random,

$$\begin{aligned}
&\Pr[A_{P_1} \cap \ldots \cap A_{P_k}] \\
&\quad = \sum_{C \in M} \Pr[E_C] \cdot \Pr[A_{P_1} \cap \ldots \cap A_{P_k} \mid E_C]
\end{aligned}$$

$$\begin{aligned}
&= \sum_{C\in M} \Pr[E_C] \prod_{i=1}^{k} \Pr[A_{P_i} \mid A_{P_1} \cap \ldots \cap A_{P_{i-1}} \cap E_C] \\
&= \sum_{C\in M} \Pr[E_C] \prod_{i=1}^{k} \left(\sum_{C'\in M_i} \Pr[E_{C'} \mid A_{P_1} \cap \ldots \cap A_{P_{i-1}} \cap E_C] \cdot \Pr[A_{P_i} \mid E_{C'}] \right) \\
&\leq \sum_{C\in M} \Pr[E_C] \prod_{i=1}^{k} \left(\sum_{C'\in M_i} \Pr[E_{C'} \mid A_{P_1} \cap \ldots \cap A_{P_{i-1}} \cap E_C] \cdot p \right) = p^k \,.
\end{aligned}$$

Hence, for an upper bound on the number of unsuccessful, non-dumped packets the events A_{P_i} can be treated as if independent with a probability of p to be true. Thus, using the Chernoff bounds, the probability that segment q has an edge that is contained in at most $2(\beta T)^2 \log T$ active segments of unsuccessful, non-dumped packets, where $\beta^2/3 \geq c+1$, is at most

$$T^3 \cdot \mathrm{e}^{-((\beta T)^2 \log T)/3} \leq \mathrm{e}^{3 \ln T - (c+1) T^2 \log T} \leq \mathrm{e}^{-cT^2 \log T}$$

if T is sufficiently large. □

Lemma 5.8 *Consider any $r' \leq r$ and packet P with a path in $S_{r'}$, that was unsuccessful in Phase 1 but was not dumped. If the congestion bound in Lemma 5.7 is true for P's path then it holds: the probability that P is unsuccessful in Phase 3 is at most $\mathrm{e}^{-cT\cdot \log T}$ for any constant $c > 0$ depending on b.*

Proof. Let β be defined as in the proof of Lemma 5.4. Since the packets choose a random delay out of a range of $[2(\beta T)^2]$, and at most $2(\beta T)^2 \log T$ active segments cross any edge, the expected number of packets that aim for one site is at most $\log T$. Hence, together with the Chernoff bounds, the probability that P is not successful in one subpass is at most

$$T^2 \cdot \left(\frac{\mathrm{e}^{\beta-1}}{\beta^{\beta}} \right)^{\log T} \leq T^2 \cdot \mathrm{e}^{-\beta \log T} \leq \mathrm{e}^{-(\beta-2)\log T}$$

for $\beta \geq \mathrm{e}^2$. Thus, for $\beta \geq \max[c+4, \mathrm{e}^2]$ the probability that P fails in at least T of $2T$ subpasses is at most

$$\binom{2T}{T} \cdot \mathrm{e}^{-(\beta-2)T\log T} \leq \mathrm{e}^{2T-(\beta-2)T \log T} \leq \mathrm{e}^{-cT \log T} \,.$$

□

Hence, the probability that some fixed packet with path in $S_{r'}$ $(r' \leq r)$ fails in round r is at most

$$\underbrace{\mathrm{e}^{-cT}}_{\text{Lemma 5.4}} + \underbrace{\mathrm{e}^{-cT^2\cdot \log T}}_{\text{Lemma 5.7}} + \underbrace{\mathrm{e}^{-cT\cdot \log T}}_{\text{Lemma 5.8}} \leq 2^{-cT} \,,$$

which completes the proof of item (2) at the beginning of this section. Since β is a constant, Phase 1, 2, and 3 each take $O(T^3 \log T) = O(2^r \cdot r)$ time steps. From this item (1) follows, which completes the analysis of the protocol.

5.2.2 The SLP Protocol

In this section we transform the FLP protocol into a protocol called SLP with an expected packet delay of $O(c_p + d_p \cdot \min[\log(c_p + d_p), \log\log n])$ steps, where n is the size of the network. As before, we consider all paths to be partitioned into sets $S_1, S_2, \ldots$ If $i \leq 3 \log\log n$, S_i contains all paths p with $\max[c_p/i, d_p] \in [2^{i-1}, 2^i)$. For $i > 3\log\log n$, S_i contains all paths p with $\max[\frac{c_p}{\gamma \log\log n}, d_p] \in [2^{i-1}, 2^i)$. Again, our protocol works in *rounds*. For every $r \geq 1$, round r is responsible for sending all packets with paths in S_r to their destinations.

Rounds 1 to $r' = 3\log\log n$ work as in the FLP protocol. For round $r > r'$, the paths of all participating packets are divided into *path segments* of length $\log^3 n$. Round r consists of $3 \cdot 2^{r-r'}$ subrounds that work like round r' in the FLP protocol. Each of these subrounds is responsible for the switching of the packets along one of their path segments. Initially, every packet chooses a *global delay* Δ i.u.r. from the range $[2 \cdot 2^{r-r'}]$. A packet with a global delay of Δ waits for Δ subrounds and afterwards participates in $2^{r-r'}$ consecutive subrounds. If, for some $i \geq 1$, it fails to cross its ith path segment in subround $\Delta + i$, it gets dumped, i.e., it remains at its current edge buffer until the end of round r.

Since rounds 1 to r' work as in the FLP protocol, it remains to consider those rounds r with $r > r'$ in order to prove Theorem 5.2. Consider some packet P with a path q belonging to some $S_{r''}$ that is currently in round $r > r'$ with $r \geq r''$. According to the SLP protocol, the expected congestion at any fixed edge in q within any fixed subround of round r is at most

$$\frac{r' \cdot 2^r}{2 \cdot 2^{r-r'}} = \tfrac{1}{2} r' 2^{r'} = \tfrac{1}{2} \cdot 3(\log n)^3 \log\log n \; .$$

Since q can have a length of at most n^2 (the maximum number of edges in the system) to be loop-free, and since the packets choose their global delays independently at random, the Chernoff bounds imply that the congestion in every subround in which P participates is at most $r'2^{r'}$, w.h.p. (i.e., with a probability of at least $1 - 1/n^\alpha$ for any constant $\alpha > 0$ if n is sufficiently large). Thus, w.h.p., every subround of P fulfills the requirements for a packet to have a path in $S_{r''}$ for some $r'' \leq r'$. Hence, the analysis of the FLP protocol implies that P successfully participates in each subround, w.h.p. Obviously, the runtime of P for the case that it fails in none of its subrounds in round $r > r'$ is at most

$$O(2^{r-r'} \cdot (r' \cdot 2^{r'})) = O(r' \cdot 2^r) = O(2^r \cdot \log\log n) = O(c_p + d_p \cdot \log\log n)) \; .$$

Combining this with a probability of at most $1/n^\alpha$ for any constant $\alpha > 0$ that P fails yields Theorem 5.2.

5.3 Dynamic Packet Switching

In this section we prove Theorem 5.3. For this, we first show how to modify the SLP protocol to obtain an $O(c_p + d_p \cdot \log\log n)$-protocol for the dynamic packet switching model, called *FDLP protocol*.

5.3.1 The FDLP Protocol

In the following, we call the time step at which a packet is injected its *birth date*. We define a path p to belong to set S_i if, at the time of its injection, $\max[\frac{c_p}{2\log\log n}, d_p] \in [2^{i-1}, 2^i)$. (Recall that c_p is the congestion caused by all packets that are at least as old as the packet following p.) One of the differences to the static situation is that we now assume that all edges have a bandwidth of $3\beta \log\log n$. The available bandwidth is divided into three parts. Each part has a bandwidth of $b = \beta \log\log n$. The ith part is reserved for packets that are in phase i of their current round of the FDLP protocol (see below). As noted in Section 5.2.1, each step using edges of this bandwidth can be simulated by $3\beta \log\log n$ steps using edges of bandwidth 1.

We assume that all paths used by the packets are short-cut free. As we will see, due to this assumption the FDLP protocol does not require ghost packets any more. The protocol works as follows:

Every newly injected packet starts *immediately* with round 1. As soon as it finishes round r, it starts with round $r+1$, and so on, until it reaches its destination. For $r \leq 2\log\log n$, round r works as follows:

Round r of the FDLP protocol for packet P:

- **Phase 1:** This phase takes $3T^2$ time steps. If P has not yet reached its destination, it chooses i.u.r. an initial delay $\delta \in [2T^2]$ with $T = \sqrt{2^r}$. If P chooses a delay of δ, it waits in its current edge buffer for δ time steps and then tries to move on without waiting until it reaches its destination or traversed T^2 edges.

  In the case that P wants to cross an edge e at the same time as at least b other packets that are at least as old as P, then P remains in the outgoing buffer of e until the end of the phase. Otherwise, P tries to reserve bandwidth for its counting packet for exactly $3T^2$ time steps later. If this is not possible, because at that time step the bandwidth of e has already been completely reserved by b non-younger packets, then P also remains in the outgoing buffer of e until the end of the phase. In both cases, P is called *unsuccessful*.

- **Phase 2:** This phase also takes $3T^2$ time steps. For packet P, a counting packet P' is sent along P's path, using the same initial delay as P. Since P successfully reserved bandwidth for P' till its current position, P' will manage to reach P in Phase 2. The task of P' is to count the number of possible delays for P for which either P or P' would have been unsuccessful (i.e., P' has to count the number of delays for P for which either the run of P or the run of P' contains a site with at least b non-younger packets, not counting P and P'). If P' counts more than $\min[\beta T/\log T, \beta \log n/\log\log n]$ of these delays, P is dumped, which means that it stays in its current edge buffer for the rest of the round. Obviously, P' needs only $O(\log n)$ bits to store the necessary information.

- **Phase 3:** The third phase consists of $2T$ *passes*. If P was unsuccessful but not dumped, then P chooses for every pass i.u.r. an initial delay from a range of $[\gamma T]$, where γ is a sufficiently large constant depending on β. Then it attempts to move along the next T edges of its path without waiting. If P intends to traverse some edge together with at least b non-younger packets at the same time, then P stops for this pass. Once P reaches its destination or traversed altogether T^2 edges in the current round, it stops and is declared *successful*.

For $r > 2\log\log n$, round r works as round r of the SLP protocol with the difference that $r' = 2\log\log n$ and a subround works as round r' above.

Since we restrict ourselves to short-cut free paths, the probabilities that different runs have a $\geq b$-site are negatively correlated and therefore can be regarded as independent of each other for an upper bound on these runs. (Thus, we do not need ghost packets any more to reduce dependencies.) As we will see, this can be used to prove that, for any round r of a packet whose path belongs to $S_{r'}$ with $r' \leq r$, the probability that this packet is not successful is at most $\mathrm{e}^{-c\cdot\min[T,\log n]}$ for any constant $c > 0$. Since round r now takes only $O(T^2)$ steps (instead of $O(T^3)$ steps for FLP and SLP), the probability for an additive delay of at least t can be improved to at most $2^{-\Theta(\sqrt{t/\log t})}$. We will prove this by three lemmas. In the following, let the function ϕ be defined as $\phi(x) = \min[x, \log n]$.

Lemma 5.9 *If the path of a packet P belongs to $S_{r'}$ for some $r' \leq r$, then the probability that it is dumped in round r is at most $\mathrm{e}^{-c\cdot\phi(T)}$ for any constant c depending on β.*

Proof. Suppose that P is in round $r \geq r'$. If $r \leq 2\log\log n$, then P has to follow a path of length $T^2 = 2^r$ and the congestion along this path caused by packets that are at least of the same age as P is at most $r \cdot 2^r = 2T^2 \log T$. Clearly, each of these packets has to be in a round of number at least r. Thus, the range from which these packets have chosen their most recent delay is at least $2T^2$. Since the delays are chosen independently at random, the probability that $b = \beta \log\log n$ of these packets (resp. their counting packets) aim for some fixed site in the run of P (resp. P') is at most

$$\binom{2T^2 \log T}{b} \left(\frac{1}{2T^2}\right)^b \leq \left(\frac{\mathrm{e}}{\beta}\right)^{\beta \log T} .$$

Let $X_1, \ldots, X_{2T^2}$ denote a collection of binary random variables with the property that X_i is 1 if and only if the ith run of P or the ith run of P' has at least one $\geq b$-site. Furthermore, let $X = \sum_{i=1}^{2T^2} X_i$. Recall that Phases 1 and 2 have their own bandwidth. Thus, packets and counting packets cannot interfere with each other. Hence, for each X_i it holds that

$$\Pr[X_i = 1] \leq T^2 \cdot 2 \left(\frac{\mathrm{e}}{\beta}\right)^{\beta \log T} .$$

If $\beta \geq 2\mathrm{e}$ and T is large enough, then

$$\mathrm{E}[X] \;\leq\; \sum_{i=1}^{2T^2} 2T^2 \cdot \left(\frac{\mathrm{e}}{\beta}\right)^{\beta \log T} \leq 4T^4 \cdot T^{-\beta} \leq 1 .$$

Since we only allow short-cut free paths, the X_i are negatively correlated. Hence, we can use the Chernoff bounds to bound the probability that at least $\beta T/\log T$ of P's runs have a $\geq b$-site by

$$\Pr[X \geq \beta T/\log T] \leq \left(\frac{\mathrm{e}}{\beta T/\log T}\right)^{\beta T/\log T} \leq \mathrm{e}^{-\beta T/2}$$

if T is sufficiently large and $\beta \geq \mathrm{e}$. Thus, the probability that P is dumped is at most e^{-cT} with $c = \beta/2$.

If $r > 2\log\log n$, then it can be shown in a similar way as above (see the analysis of the SLP protocol) that the probability that P is dumped is at most $\mathrm{e}^{-c\log n}$. □

Lemma 5.10 *Consider any packet P with path in $S_{r'}$ that is in Phase 3 of round $r \geq r'$. The probability that there is an edge along P's path with a congestion of at least γT is at most $\mathrm{e}^{-c\cdot\phi(T)}$ for any constant $c > 0$ depending on β.*

Proof. Recall that Phase 3 has its extra part of the bandwidth. Hence, the congestion we have to bound is solely due to packets that are also in Phase 3 of some round.

Consider some fixed edge e along the path of P. First, we bound the expected number of active path segments of packets participating in Phase 3 that contain e. Suppose that $r \leq 2\log\log n$. We know that there are at most $r \cdot 2^r = 2T^2 \log T$ packets in the system that intend to cross e and are at least as old as P. Since P is already in Phase 3 of its round, this means that all the other packets must be in Phase 3 of some round at least r in order to contribute to the congestion. Let the random variable X denote the congestion caused by these packets.

Similar to Lemma 5.7, it can be shown that the probability of a packet to be unsuccessful and not dumped is at most $(\beta T/\log T)/(2T^2)$. Thus,

$$\mathrm{E}[X] \leq 2T^2 \log T \cdot \frac{\beta T/\log T}{2T^2} = \beta T \ .$$

Since, according to the proof of Lemma 5.7, the probability of a packet to be unsuccessful and not dumped can be viewed as being independent of the other packets for an upper bound on the congestion, we can use the Chernoff bounds to obtain

$$\Pr[X \geq 2\beta T] \leq \mathrm{e}^{-\beta T/3} \ .$$

Thus, the probability that there is an edge along P's path with congestion at least $2\beta T$ is at most

$$T^2 \cdot \mathrm{e}^{-\beta T/3} \leq \mathrm{e}^{-cT}$$

if $\beta \geq 3c+1$ and T is sufficiently large.

For $r > 2\log\log n$, a similar proof shows that the probability that P has a congestion of at least $2\beta \log n$ along its path is at most $\mathrm{e}^{-c\log n}$. □

Lemma 5.11 *Consider any packet P with a path in $S_{r'}$ that was unsuccessful in Phase 1 of round $r \geq r'$ but was not dumped. If the congestion bound in Lemma 5.10 is true for P's path, then it holds: the probability that P is unsuccessful in Phase 3 is at most $\mathrm{e}^{-c\cdot\phi(T)}$ for any constant $c > 0$ depending on β.*

Proof. The proof of this lemma is analogous to the proof of Lemma 5.8. □

The three lemmas above imply that the probability of a packet with path in $S_{r'}$ to fail a round $r \geq r'$ is at most $2^{-c\cdot\phi(T)}$ for any constant $c > 0$, as desired. Next we show how to transform the FDLP protocol into a dynamic protocol, called DLP, that is stable.

5.3.2 Stability Under Any Bounded Injection Model

In order to complete the proof of Theorem 5.3, it remains to be shown how to ensure stability. Consider an arbitrary path selection strategy based on an arbitrary bounded injection model and applied to an arbitrary network. Let T be the minimum number for which there is a λ such that, for any fixed time interval of length T and edge e, at most λT paths are injected (w.h.p.), that contain e. Furthermore, set $D = d_{\max} \cdot \log\log n$ with $d_{\max}$ being the length of a longest possible path injected into the system.

We extend our dynamic protocol presented in the previous section to a protocol called DLP in a way similar to our black box transformation described in the previous chapter: we use SIS as a primary contention resolution rule for all packets. All packets that are not longer in the system than some threshold t (to be determined later) are assumed to have age 0. These packets are routed according to the FDLP protocol. All other packets have an age determined by their injection time. To these packets, the pure SIS protocol is applied.

We already know that the time needed by the FDLP protocol to send a packet along a path p is at most $\alpha \cdot (c_p + d_p \cdot \log\log n) + \beta \log^2 n$, w.h.p., where α and β are constants. Let t be chosen as $\max[\lceil 2(\alpha D + \beta \log^2 n)/T \rceil,\ 1] \cdot T$. Since the congestion of a newly injected path

p due to packets that are at most t steps old is at most $\lambda \cdot t$ (w.h.p.) according to our injection model, the time a packet needs to traverse p is at most

$$\begin{aligned} \alpha \cdot (\lambda \cdot t + D) + \beta \log^2 n &\leq \alpha \cdot \left(\lambda \cdot t + \frac{t}{2\alpha} - \frac{\beta \log^2 n}{\alpha} \right) + \beta \log^2 n \\ &= (\alpha\lambda + 1/2) \cdot t \end{aligned}$$

w.h.p. This is less than t if $\lambda < \frac{1}{2\alpha}$. Hence, it follows from the analysis of the black box transformation scheme in the previous chapter that for any constant $\lambda < \frac{1}{2\alpha}$ the DLP protocol is stable and the expected routing time of a packet is the same as in the FDLP protocol. This concludes the proof of Theorem 5.3.

5.4 Conclusion

In this chapter, we presented locally efficient online switching protocols for static and dynamic packet injections. Many open problems remain. For instance, is it possible in our local models to route a packet online along a path p in expected time $O(c_p + d_p)$? Can our results for the dynamic injection model be generalized to arbitrary simple paths? Is it possible to avoid control packets for locally optimal switching protocols? Our switching protocols may not run efficiently in practice. Are there local switching protocols that work well both in theory and in practice?

Chapter 6

Routing in Ad Hoc Wireless Networks

An *ad hoc* wireless network is a collection of wireless mobile hosts forming a temporary network without the aid of any established infrastructure or centralized administration. This type of network is of great importance in situations where it is very difficult to provide the necessary infrastructure, but it is a challenging task to enable fast and reliable communication within such a network. In this chapter, we model and analyze the performance of so-called *power-controlled* ad hoc wireless networks: networks where the mobile hosts are able to change their transmission power. We concentrate on finding schemes for routing arbitrary permutations in these networks. In general, it is NP-hard even to find a $n^{1-\epsilon}$-approximation for any constant ϵ to the fastest possible strategy for routing a given permutation problem on n mobile hosts. However, we demonstrate here that if we allow ourselves to consider slightly less general problems, efficient solutions can be found.

We first demonstrate that there is a natural class of distributed schemes for handling node-to-node communication on top of which online path selection and switching strategies can be constructed such that the performance of this class of schemes can be exploited in a nearly optimal way for routing permutations in any static, power-controlled ad hoc network. We then demonstrate that if we restrict ourselves to the important case of routing between nodes distributed randomly in a Euclidean space, we can route in a time that is asymptotically optimal for any routing scheme.

6.1 Introduction

At the moment a dramatic growth in wireless networks can be observed. Mobile hosts and wireless networking hardware are becoming widely available, and extensive work has been done recently in integrating these elements into traditional networks such as the Internet. There are, however, important scenarios in which no fixed wired infrastructure such as the Internet is available, either because it may not be economically practical or physically possible to provide the necessary infrastructure, or because the expediency of the situation does not permit its installation (for instance, networks formed by satellites, ships or airplanes, or networks connecting rescue teams in case of an earthquake or flood). In such situations, a collection of mobile hosts with wireless network interfaces may form a temporary network without the aid of any established infrastructure or centralized administration. This type of wireless network is known as an *ad hoc* network [Gib96, IE96]. Ad hoc networks have been recognized as an important form of wireless network. The IEEE 802.11 recommended standard for wireless LANs, for instance, requires an access protocol to have the ability to support ad hoc networking [Che94].

Ad hoc networks can be divided into two categories: *simple*, where the mobile hosts use fixed transmission powers, and *power-controlled*, where the mobile hosts are able to change their transmission power. Power-controlled networks have several advantages over simple ad hoc networks:

- Contention among the hosts can be significantly reduced.
- Energy consumption can be significantly reduced.
- Security of transmissions can be increased.

The inflexibility of simple ad hoc networks can be best demonstrated by considering human conversation. In everyday life we naturally use our ability to control our voice level in order to communicate with other people. Imagine the problems caused if everyone could only speak with one voice level!

In this chapter, we develop strategies for the efficient exchange of messages in power-controlled ad hoc networks. The most important communication primitives for these networks are broadcasting and point-to-point communication. We here concentrate on coordinating sets of point-to-point communication problems that can be described as permutations. The majority of traffic on the Internet today consists of point-to-point communication, and it is reasonable to assume that traffic on ad hoc networks will be similar. However, to the best of our knowledge, no strategy or algorithm has been presented previous to our work that can solve permutation problems by a method better than a sequence of broadcasts. This is a waste of communication resources, which is undesirable, since bandwidth in wireless networks is very scarce compared to wired networks. Instead, we here demonstrate much more time- and resource-efficient techniques that are close to optimal in certain situations.

6.1.1 Previous Work

Despite the advantages of allowing variable transmission powers, much of the existing work on ad hoc networks considers hosts that have a fixed transmission range. However, the idea of allowing mobile hosts to vary their transmission power and transmission rate is already used in so-called power-controlled CDMA systems, where the base station can direct mobiles to reduce their power and data rate to reduce interference and allow more users on the system. This approach is employed in TIA IS-95 with respect to the time-varying voice activity on cellular voice channels [Jac96, Gib96]. The advantage of adjusting the power and data rate of mobile data users to the current interference level has already been studied experimentally (see, e.g., [Jac96]).

To our knowledge, no results for an abstract model exist on how to select routes or schedule the transmissions of messages for power-controlled ad hoc networks. However, connectivity problems have been studied for both simple ad hoc networks [Pir91], as well as power-controlled ad hoc networks [UY96, KKKP97]. Kirousis et al. [KKKP97] present a polynomial time algorithm for finding the minimum cost transmission power assignment that maintains connectivity for arbitrarily distributed collinear points, where the cost of an assignment is defined as the total power usage. They also provide a 2-approximation algorithm for finding the minimum cost assignment in the case where the hosts are arbitrarily distributed in three dimensions.

The remainder of the previous work we discuss has studied only simple ad hoc networks. A common model for such networks is called the *packet radio network* (or PRN) model. A PRN is represented as an undirected graph $G = (V, E)$, where two nodes i and j are connected if and only if i is within the transmission radius of j and vice versa. One time step is defined as the time it takes to transmit a packet. Since it is usually assumed that all nodes can only transmit at one frequency, a *transmission conflict* occurs at node i if two of its neighboring nodes want to transmit at the same time. Communication problems have been studied by many authors for both the case that a conflict can be detected and the case that it cannot be detected.

Studies of multi-hop PRNs have mainly concentrated on broadcasting problems. Lower bounds on this problem are considered in [ABLP91, KM93, BGI92, Hwa95, BP97]. A number of papers also deal with constructing efficient broadcast protocols [CK85, Wei87, CW91, BGI92, GM95, KR95]. Let n denote the size, D denote the diameter, and Δ denote the maximum degree of a given PRN. Bar-Yehuda et al. [BGI92], for instance, present a randomized distributed broadcast protocol that completes in expected $O(D \log n + \log^2 n)$ steps. Gaber and Mansour [GM95] present a centralized deterministic protocol that works in time $O(D + \log^5 n)$. Point-to-point communication problems have also been studied, for instance, by Bar-Yehuda et al. [BII93], who present a randomized distributed protocol that performs k point-to-point transmissions in $O((k + D) \log \Delta)$ time steps, on average.

Another problem that has been studied is how to schedule transmissions in a PRN to enable neighboring nodes to successfully exchange information. For early work, see e.g. [CP87, CS89, PSK85]. Recent results can be found in [CF94, Boz95]. Also, the problem of selecting paths and updating paths when mobile hosts move has been considered. See, e.g., [MJP95, JM96, DSB97] for different route selection strategies and further references. Similar problems have been studied for various dynamic network models. See Dolev et al. [DKKP95] for a survey of results in this area.

6.1.2 The Model

It is a challenging task to develop a model for wireless communication that is simple enough to enable the design and analysis of algorithms, but is also detailed enough to ensure that efficient algorithms derived in this model also perform well in practice. As in previous approaches such as the PRN model, we choose a high level of abstraction here. We also note at the end of this section additional details that could be added to the model, and how they would affect our results.

We model power-controlled ad hoc networks with the help of the following graph:

Definition 6.1 *Let a* transmission graph $\mathcal{G} = (V, \tau)$ *be a complete undirected graph with node set V and edge labels determined by the function $\tau : V \times V \to \mathbb{R}_+$. For any edge $\{u, v\}$, $\tau(\{u, v\})$ represents the lowest transmission power that allows u to send a message to v and vice versa.*

In this chapter, we only deal with *static* transmission graphs, i.e., the situation where the positions of the mobile hosts and the environment do not change. Although dynamic transmission graphs are more interesting for many wireless networking problems, static graphs are useful for scenarios where a large amount of communication can be performed between updates to the transmission graph. This is the case for fixed wireless networks in buildings or connecting several buildings, or low-mobility radio networks such as networks formed by ships. A thorough understanding of static graphs also provides a starting point for the dynamic case, for example, by providing a measure with which to compare performance.

The remainder of the model is defined as follows:

- A packet needs one time step to make a single hop in the network, regardless of the distance to the destination node. In many situations, it is more time- and resource-efficient for a message to perform a sequence of hops instead of one single hop to its final destination. The sequence of nodes used for the hops is called the *route* of a message.

- Only one frequency is available for transmitting packets, which implies that:

  - A node can send out at most one packet at a time.
  - If a node v attempts to send a packet with transmission power t, then all nodes that need less than αt power to receive a packet from v are *blocked*, where $\alpha > 1$ is some fixed constant. Any information transmitted to a blocked node is not received. Note that this means that it is possible for no transmission during a single time step to be successful.

- A transmission conflict cannot be detected by the sending node.

- All nodes work in a synchronized way. (For simplification reasons, this paradigm is commonly used in the design and analysis of wireless communication strategies that are robust to slight differences in the speeds at which the nodes operate [BGI92, Gal85, Rob72].)

As noted above, the level of abstraction of this model is high. We here address three of the above assumptions: (a) all hops in the network take the same amount of time, (b) the cause of messages being blocked, and (c) the maximum transmission power of the mobile stations is unlimited. For (a), in practice, signal propagation time does cause the time required for a hop to depend on the distance traveled. However, even though incorporating this into our model does make our proofs slightly more complicated, it does not change any of our results in a qualitative manner. For (b), we point out that in practice, a message m can be blocked even when no single node is transmitting with enough power to block m, but the cumulative effect of the transmissions by several nodes might block m. The relevant measure is actually the strength of the interference caused by all possible sources of signals (the so-called *signal to interference ratio* or SIR) and not only one. See, for instance, the model developed by Ulukus and Yates [UY96]. However, in practice it turns out that only signals with strength above some threshold value contribute to blocking a node, since all other signals tend to cancel each other out rather than to add up, or may even be insignificant compared to the white Gaussian noise that is always present. Furthermore, incorporating the SIR into our model in the manner proposed by [UY96] makes our proofs considerably more complicated, but has no qualitative effect on the results of Section 6.2 and only an insignificant qualitative effect on the results of Section 6.3. For (c), limitations on the transmission power can be easily incorporated in Section 6.2 by restricting the class of schemes considered accordingly. This also does not present any difficulties for the strategies considered in Section 6.3, since they only require communication between nodes of close proximity.

6.1.3 On the Hardness of Routing in Ad Hoc Networks

Several authors have already presented NP-hardness results for various wireless communication problems, such as the problem of finding an optimal broadcast schedule [CK85] or scheduling transmissions in the case where every node wants to send a message to one of its neighbors [SH96]. We show the following similar result:

Remark 6.2 *Given a transmission graph and a collection of n paths, finding a schedule that completes in a number of time steps that is within a factor of $n^{1-\epsilon}$ of optimal, for any constant $\epsilon > 0$, is NP-hard.*

Proof. (Sketch) This is by a reduction from Chromatic Number, which is not approximable to within a factor of $n^{1-\epsilon}$ unless NP = coRP (which is widely believed not to be true) [FK96]. Given a graph G for which we wish to find an approximation to Chromatic number, we construct a transmission graph G' as follows: for every node $v \in G$, we have two nodes $v'_1, v'_2 \in G'$. The label on the edge between v'_1 and v'_2 is 1. Also, if there is an edge between nodes u and v in G, then the label on the edges between u'_1 and v'_2 as well as the edge between v'_1 and u'_2, are both $\alpha^{1/2}$. The remainder of the labels on edges in G' are α. The routing task is to transmit a single message from each node v'_1 directly to the corresponding v'_2. It is easy to show that any schedule of length T for the problem in graph G' provides a T-coloring for G, and that any T-coloring for G provides a schedule of length T for the problem in graph G'. □

Hence, finding efficient communication strategies for wireless networks appears to be much harder than for networks with fixed node-to-node connections, even if a fixed collection of paths is already given along which the packets have to be sent. A similar claim also holds for the case where we do not constrain ourselves to specific paths. The details are slightly different, but the reduction in the proof is the same, since a transmission to node v'_2 from any other node besides node v'_1 blocks every other node in the transmission graph.

6.1.4 New Results

Nearly nothing is known about how permutation routing problems can be handled in wireless communication networks, and the hardness results above do not look very promising. However, we show here that if we allow ourselves to consider slightly less general problems, efficient solutions can be found. We demonstrate this with two approaches: first, we consider the effect of restricting the class of routing strategies allowed, and then we restrict the class of allowed transmission graphs. Both restrictions we consider are natural and lead to permutation routing strategies that are close to optimal for their respective classes.

The class of strategies considered is defined by separating the routing problem into three layers: one layer that handles the scheduling of node-to-node transmissions of packets (we follow the experimental literature and refer to this as the *medium access control*, or MAC, layer [Gib96]), one layer that is responsible for finding efficient paths for the packets (called the *path selection* layer), and one layer for selecting which packets are to be transmitted next by the nodes (called the *switching* layer). We introduce a class of protocols for the MAC layer called *local probabilistic control* protocols. These protocols allow every node to operate without any coordination among the nodes. Furthermore, they ensure that each attempted transmission is successful with constant probability.

On top of these protocols we develop schemes for the other two layers that almost optimally exploit the power of the MAC layer protocols for *any* transmission graph. For this we introduce a parameter $R(G, S)$, similar to the routing number defined in Example 2 of Section 3.2.5, that represents a lower bound on the expected time required to route a random permutation in G if restricted to the MAC scheme S and to pure switching protocols (that is, only one copy per packet is allowed to be sent). We present online strategies for the path selection layer and switching layer that are able to route any permutation in G using S in time $O(R(G, S) \cdot \log n)$, and in many cases even in time $O(R(G, S) \cdot (\log \log n)^2)$, where n is the size of G. The techniques developed include efficient simulations of strategies for deterministic communication models by probabilistic communication models. This might be of independent interest.

We then study the effect of restricting the class of allowed communication graphs to those representing mobile hosts distributed in a Euclidean space. We assume that n points are distributed uniformly and independently at random in a *domain space*, a convex region of $\mathbb{R}^d$ that is known *a priori*. This random distribution models the situation where each of the nodes moves within the domain space independently and with equal probability in any direction. (To see why this is true, we refer to the Fundamental Theorem of Markov Chains that can be found, e.g., in [MR95], p. 132.)

In the case where the domain space is a 2 dimensional square, we take advantage of the considerable similarity between randomly distributed nodes in such a domain space and existing work on computing with faulty arrays [Rag89, KKL$^+$90, CMS97] to show that we can route any permutation online in time $O(\sqrt{n})$ with probability at least $1 - O(1/n)$. We also demonstrate that the techniques for a square domain space can be extended to an arbitrarily shaped convex domain space.

Furthermore, we show that in the scenario considered, it is not possible to use the ability to transmit over long distances to perform permutation routing faster using wireless communication than it is using an array: we prove a lower bound demonstrating that when the domain space is a 2 dimensional square, the time required to route a random permutation is $\Omega(\sqrt{n})$ with probability at least $1 - O(1/n)$. We also show that the result for routing in an arbitrarily shaped convex domain space is within a constant factor of optimal.

Finally, we note that our routing strategies developed for the second approach above (i.e., restricting the class of allowed transmission graphs) are not limited to supporting the exchange of messages, but can also be used to perform efficient distributed computations in ad hoc networks such as sorting and matrix multiplication.

6.2 A Universal Framework for Wireless Communication

In this section, we propose strategies that are universally applicable in a sense that they can be reliably used for arbitrary static ad hoc networks and arbitrary routing problems. Our approach basically consists of three layers:

(1) the medium access control layer (or MAC layer),

(2) the path selection layer, and

(3) the switching layer.

The MAC layer is responsible for enabling node-to-node communication. The task of the path selection layer is to choose suitable paths (i.e., suitable sequences of nodes to visit) for the packets, and the task of the switching layer is to resolve conflicts between packets that want to be transmitted by the same node at the same time step.

A communication scheme has to fulfill several demands to be strictly online, such as: the paths should be chosen independently of each other, and the nodes should be able to decide locally when and how often to try to get access to other nodes within some given time interval, and which packet to prefer if several packets want to be sent out by a node at the same time step. On the other hand, the strategy should (at least in important scenarios) ensure that the node-to-node communication, the selection of the paths, and the switching are as efficient as possible. As we have seen in the previous section, to fulfill even a part of these demands is impossible in general (unless NP = coRP). However, we will show that if we restrict ourselves to considering only communication strategies that use the class of MAC layer protocols described below, suitable strategies can be constructed that (nearly) fulfill these demands.

6.2.1 Strategies for the MAC Layer

In this section, we propose a class of local probabilistic control protocols for the MAC layer (also called *LPC schemes* in the following) that allow all nodes to operate independently of each other without blocking any other node too much by access trials. Before we define this class, let us first give a brief overview of the most commonly used MAC layer strategies and describe, why they cannot be used efficiently in ad hoc networks.

Currently used MAC layer protocols

In todays wireless communication technologies, there are basically three different ways of transmitting signals: FDMA, TDMA and CDMA.

In FDMA (frequency division multiple access), signals can be transmitted simultaneously by using different frequencies. In TDMA (time division multiple access), signals occupy the same frequency band, but are separated by using non-overlapping time slots for transmissions. CDMA (code division multiple access) is a technique that contains both FDMA and TDMA. The basic idea of CDMA is that the senders use codes for their transmissions that are orthogonal to each other, i.e., their cross-correlation is 0. Thus, with the help of a suitable correlator a receiver is able to filter out a signal from a specific sender without getting interference from others.

All current access schemes use either TDMA or CDMA (both often in combination with FDMA to increase the bandwidth and/or reduce interference). Whereas TDMA is used in most of todays standards such as DECT, GSM (Europe), PDC (Japan) and IS-54 (USA), CDMA is regarded as the access mode for future wireless communication standards such as UMTS (universal mobile communication system), currently under development in Europe, or IMT-2000, currently under development by the ITU, the international body for communication standards.

Both TDMA and CDMA are not useful for ad hoc wireless networks since they are very sensitive to even small differences in the speeds at which the mobile stations operate. When there is central control, for instance via a base station, this problem is solved with the help of a reference signal sent out by the base station at fixed time intervals.

In the special case of wireless LANs, most networks are based on carrier sense multiple access (CSMA), polling, and TDMA. These can also be found in the IEEE 802.11 recommended standard for wireless LANs.

In CSMA protocols, each node listens to whether other nodes are transmitting. If not, it tries to send its message. If it recognizes a collision with another message, it backs off for a random period of time before it tries again to send out its message. CSMA has been used with great success in the Ethernet. Unfortunately, this strategy is not applicable for ad hoc networks, since in general a node cannot detect whether the destination of its message is currently blocked or not. (This problem is known as the *hidden terminal problem* and has received considerable attention in the ad hoc network community.)

Hence, we propose a different class of MAC layer protocols for ad hoc networks. The advantage of these protocols is that they seem to have the potential to adjust quickly to changing situations and therefore could support efficient communication also in dynamic ad hoc networks.

A new class of MAC layer protocols

In this section, we propose a class of simple local probabilistic control protocols for the MAC layer that allow each node to decide independently of the other nodes when to send a packet.

Let us start with describing how to perform a hop in an ad hoc network. Suppose v tries to transmit a packet P to w. We say that the hop of P is *successful* if v knows that P reached w. Since in our model a transmission conflict cannot be detected by the sending node, a successful hop requires sending data without a conflict from v to w and from w to v. For our purposes it would suffice to have only two steps: sending P from v to w, and sending an acknowledgement from w to v. The acknowledgement is necessary, since we require v to know whether w received P. Otherwise it will go on trying to transmit P. However, we do not require w to know whether v knows that P reached it, since our path selection strategies will be based on choosing fixed paths. Once w receives the packet it can immediately start to send it further along its path.

Note that we do not consider here the possibility of transmission errors or node failures. In these cases, four communication steps may be used for a hop to ensure a correct transmission, and limits for the number of transmission attempts may be used to detect node failures.

Now, consider any transmission graph $G = (V, \tau)$. We define an LPC scheme for G to be a MAC layer protocol that can be described by an *access matrix* $P = (p_{v,w})_{v,w \in V}$ with $p_{v,w} \in [0, 1)$ for all $v, w \in V$. This access matrix is used in the following way:

Assume that some node v wants to send a packet Q to node w (which is allowed if $p_{v,w} > 0$). As long as v has not received an acknowledgment, v decides at each time step with probability $p_{v,w}$ to start a hop for Q to w.

For all $u, v \in V$ with $u \neq v$, let $b_{u,v} = \max_{w \in V}\{p_{v,w}$: a hop attempt from v to w blocks $u\}$. (Recall the definition of a node to be blocked in our model.) By convenience, we define $b_{u,u} = 0$. We demand for each LPC scheme that the $p_{v,w}$'s are set in such a way that for every $u \in V$, $\sum_{v \in V} b_{u,v} \leq 1/4$, i.e., the probability that u is blocked at some time step is at most 1/4 for any situation in which every node has at most one packet to transmit. This ensures that for any LPC scheme, no coordination among the nodes is necessary. Each node can make an arbitrary decision of where to try to send its packet, but we still have for all nodes $v, w \in V$ that

$$\Pr[\text{hop } v \to w \text{ not successful}] \leq \Pr[v \text{ blocked}] + \Pr[w \text{ blocked}] \leq 1/2 \; .$$

That is, a hop attempt has a success probability of at least 1/2.

We want to use this class of schemes as a basic building block for the rest of our communication strategies. We will demonstrate that we can exploit the power of these schemes in a nearly optimal way for routing arbitrary permutations in *any* static ad hoc network. Hence, possible inefficiencies are confined mostly to the MAC layer. An interesting open problem is finding more powerful classes of MAC layer protocols and developing efficient routing strategies on top of them.

Transforming the problem of routing in transmission graphs to routing in PCGs

The drawback of our schemes is that they are only guaranteed to work if every node tries a transmission only for *one* packet (i.e., only one packet can be active) at any time. Hence, if more than one packet is currently stored at a node, the others have to wait until the packet chosen by the node for hop attempts is successful. To see why this is a problem, consider the scenario where v_1 has $p_{1,2} = 1/10$ and $p_{1,3} = 1/1000$. If v_1 has to send a packet P_3 to v_3, it will (probably) require many attempts before P_3 is successfully sent. Say in the meantime, that v_1 receives another packet P_2, where P_2 needs to be sent to v_2. Forcing P_2 to wait until P_3 is successfully sent will probably force P_2 to wait at v_1 much longer than expected. On the other hand, giving P_2 priority over P_3 can lead to "starvation" of P_3 if many packets destined for v_2 arrive.

In order to avoid this drawback, we show in the following how to refine any LPC scheme to a scheme that allows several packets to be active at the same time, without reducing the $p_{v,w}$'s by more than a constant. This transformation not only helps to make our schemes much more flexible, but also helps to interpret the problem of routing in transmission graphs using our LPC schemes as a routing problem in a network model called PCG (see Definition 6.4). This allows us to draw on the large body of literature in the field of interconnection networks to propose suitable path selection and switching strategies. The analysis of these strategies, however, is much more involved than previous analyses because of the probabilistic and non-uniform nature of our network model.

Consider any LPC scheme S, and let P be its access matrix. For each node v and $i \geq 1$, let the *i-zone* of v contain all nodes w with $p_{v,w} \in (\frac{1}{2^{i-1}}, \frac{1}{2^i}]$. For each i-zone, let us define the zone access probability for v as $q_{v,i} = \frac{1}{2^{i+2}}$. Furthermore, for all $u, v \in V$ let $b_{u,v}$ be defined as above. Assume that we allow now each node to have an active packet for every zone. Then the probability that node v is blocked by a hop attempt of a node $w \neq v$ is at most

$$\sum_{w \in V} \sum_{i \geq \lceil \log b_{v,w}^{-1} \rceil} \frac{1}{2^{i+2}} \leq \frac{1}{2} \sum_{w \in V} b_{v,w} \leq \frac{1}{8} \, .$$

Since for every pair of nodes $v, w \in V$ we require that $p_{v,w} \leq 1/4$ in order to ensure the bounds on the blocking probability of the original LPC scheme, the lowest i for which hop attempts to an i-zone are allowed is at least 2. Thus, the probability that node v is blocked by one of its own hop attempts is at most

$$\sum_{i \geq 2} \frac{1}{2^{i+2}} \leq \frac{1}{8} \, .$$

Hence, altogether the probability that a node is blocked using the zone access scheme is at most 1/4. Thus, the probability that a hop attempt is successful is at least 1/2, as desired.

Let us call the scheme S extended by zone accesses S_Z. Furthermore, let P_Z be the access matrix of S_Z with entries $p_{v,w}^Z$ for all $v, w \in V$. The construction above implies that, for all $v, w \in V$ with $p_{v,w} \in (\frac{1}{2^{i-1}}, \frac{1}{2^i}]$, $p_{v,w}^Z = \frac{1}{2^{i+2}} > p_{v,w}/8$. Thus the following fact holds.

Fact 6.3 *For any LPC scheme S with access matrix P it holds that every entry of the access matrix P_Z of S_Z is at least $1/8$ the corresponding entry of P. However, S_Z allows one packet to participate in hop trials for each node and zone while guaranteeing that any attempted hop is still successful with probability at least $1/2$.*

In the following, we show that for all communication strategies using a zone access scheme we can transform the problem of routing packets in a transmission graph (which is very difficult to handle, as we saw in the introduction) to the problem of routing packets in the following graph model (which, as we will see, is easier to handle).

Definition 6.4 *Let the* probabilistic communication graph *(or PCG in short) $G = (V, \varphi)$ be defined as a complete directed graph with node set V and edge labels determined by the function $\varphi : V \times V \to [0, 1]$. Every edge e can forward a packet in one time step, but only succeeds in doing this with probability $\varphi(e)$.*

Next we define how to transform transmission graphs to PCGs.

Definition 6.5 *Given a transmission graph $G = (V, \tau)$ and a zone access scheme Z with maximum zone number k, let the corresponding PCG be defined as $G_Z = (W, \varphi)$ with*

- $W = V \cup (V \times \{1, \dots, k\})$,
- $\varphi(v, (v, i)) = \frac{1}{2^{i+2}}$ *for all $v \in V$ and $i \in \{1, \dots, k\}$ (this represents the probability that a hop is attempted to the i-zone of v in G),*
- $\varphi((v, i), w) = 1$ *if w is in the i-zone of v and 0 otherwise for all $v, w \in V$ and $i \in \{1, \dots, k\}$, and*
- $\varphi(v, w) = \varphi((v, i), (w, j)) = 0$ *for all $v, w \in V$ and $i, j \in \{1, \dots, k\}$.*

These transformation rules allow us to state the following result.

Theorem 6.6 *Given any transmission graph G and zone access scheme Z it holds that, for any strategy on top of Z that can route some given routing problem $\mathcal{R}$ in G in expected time T, there is an offline strategy in G_Z that routes $\mathcal{R}$ in expected time at most $2T$.*

Proof. Given a strategy S on top of Z, let $\mathcal{T}_S$ denote the probability tree of all possible outcomes for sending the packets in G to their destinations, using S on top of Z. The root of $\mathcal{T}_S$ represents the starting point, and each level at depth $t > 0$ in $\mathcal{T}_S$ represents all possible outcomes after t steps. Every edge (x, y) in $\mathcal{T}_S$ represents a possible transition from outcome x to outcome y for some time step t. For this transition, the following are all cases that can occur for any node v in G and zone z:

(1) No packet at v tried to use zone z at time t, or

(2) packet P at v tried to use zone z, but it randomly chose not to perform a hop, or

(3) packet P at v tried to perform a hop to node w in zone z, but failed due to a collision event, or

(4) packet P at v successfully performed a hop to node w in zone z.

Our aim is to use a (centralized) offline protocol for G_Z that simulates $\mathcal{T}_S$. Clearly, events of type 1 do not need any transformation (apart from, maybe, holding packets artificially back in the offline protocol). Events of type 2 can also be used without any change for G_Z, since the probability of attempting a hop is equal for both models. Since collisions of events cannot happen in G_Z, events of type 3 do not appear there. To incorporate them in the offline protocol, they are simulated in a way that the branching probabilities in $\mathcal{T}_S$ are preserved. Finally, any event of type 4 appearing in G can be taken over as it is to the offline protocol. As a hop takes two steps in G_Z instead of one step in G, the theorem follows. □

Hence, a lower bound for routing some given routing problem $\mathcal{R}$ in the PCG G_Z is also (up to a constant factor) a lower bound for routing $\mathcal{R}$ in the transmission graph G using Z. In addition, it is easy to see that for any analysis of a routing strategy, in which only probabilities for one node and zone at a time or expectations are considered for bounding the number of successful hops, any upper time bound for sending packets in the PCG G_Z also holds for sending the packets in the transmission graph G using Z. However, if probabilities for several nodes and/or zones are considered for the same time step, this is no longer true because of possible transmission conflicts. The proofs for our upper time bounds, presented in the following, will take care of this problem, so that all results presented in the following for PCGs also hold for transmission graphs using zone access schemes.

6.2.2 Path Selection Strategies for PCGs

In this section, we show how to select efficient path collections online and in a distributed way for the routing of arbitrary permutations in arbitrary PCGs. For this we need some notation. Let the *edge latency* L of a PCG G be defined as the maximum expected time $< \infty$ to cross an edge in G. Given a collection $\mathcal{P}$ of simple paths in some PCG G,

- the *dilation* D of $\mathcal{P}$ is defined as the maximum over all paths in $\mathcal{P}$ of the sum of $1/\varphi(e)$ over all edges e used by it (that is, D denotes the maximum expected time a packet needs to traverse a path in $\mathcal{P}$), and

- the *congestion* C of $\mathcal{P}$ is defined as the maximum over all edges e of $1/\varphi(e)$ times the number of paths in $\mathcal{P}$ that cross it (that is, C denotes the maximum expected time spent at an edge e to forward all packets which contain e in their path).

In order to have a measure for the quality of the dilation and congestion of a path collection, we use the so-called *routing number*. In this chapter, the routing number is defined as follows:

Consider an arbitrary PCG $G = (V, \varphi)$ with N nodes. Let S_N represent the set of all permutations over $[N]$. For a permutation $\pi \in S_N$, let $R(G, \pi)$ be the minimum possible expected number of steps required to route packets offline (i.e., the whole routing problem is known to all nodes) in G according to π under the assumption that only one copy per packet is sent through the network. Then the *routing number* $R(G)$ of G is defined by $R(G) = \max_{\pi \in S_N} R(G, \pi)$. When there is no risk of confusion about the network G, we will write R instead of $R(G)$. The routing number has the following nice property.

Claim 6.7 *For any PCG G with routing number R and any routing strategy, the average, over all permutations, expected number of steps to route a permutation in G is bounded by* $\Omega(R)$.

Proof. The proof is identical to the proof of Claim 3.15. □

Hence, asymptotically the routing number is not only an upper bound, but also a lower bound for the average permutation routing time using optimal routing strategies in G. This demonstrates that the routing number is a robust measure for the routing performance of graphs within our model. With the help of the routing number, we can prove Theorem 6.8. The main problem to be overcome in the proof of this theorem is that a best possible path selection strategy for a PCG may not be a strategy using fixed paths, but maybe a strategy in which paths are chosen adaptively according to the outcome of transmission trials. In particular, when using adaptive strategies, the expected congestion and dilation can be by far larger than the expected routing time.

Theorem 6.8 *For any PCG G with routing number R and edge latency L and any permutation routing problem* $\pi \in S_N$ *in G, there exists a collection of simple paths with congestion and dilation at most* $R + O(\sqrt{R\ell \log(R/\ell)}) = \Theta(R)$, *where* $\ell = \min[R, L]$.

Proof. Consider any strategy $\mathcal{S}$ for sending packets in G to destinations prescribed by π. Let the random variable T denote the time the packets need to reach their destinations using $\mathcal{S}$. In order to have a bound on the edge latencies used by $\mathcal{S}$, we first show the following lemma.

Lemma 6.9 *If* $\mathcal{S}$ *makes use of edges with latency beyond* $\mathrm{E}[T]$, $\mathcal{S}$ *can be reduced to a strategy* $\mathcal{S}'$ *with an expected runtime of at most* $2\mathrm{E}[T]$ *that makes use of edges with latency at most* $2\mathrm{E}[T]$.

Proof. Consider some fixed packet P, and let the random variable T_P denote the time required by P to reach its destination. Since $T = \max_P T_P$, it holds that $\mathrm{E}[T_P] \leq \mathrm{E}[T]$. For any $t \geq 1$, let the binary random variable A_t be 1 if and only if P has not reached its destination at step t, and let the binary random variable S_t be 1 if and only if P would have been successful to traverse an edge of latency at least $2\mathrm{E}[T]$ at step t if it had tried it. Furthermore, let the random variable X_t be equal to $A_t \cdot S_t$ and $X = \sum_{t \geq 1} X_t$. It is not difficult to verify that X is an upper bound for the number of successful traversals of edges of latency at least $2\mathrm{E}[T]$ by P. It holds that

$$\mathrm{E}[T_P] = \mathrm{E}\left[\sum_{t \geq 1} A_t\right] = \sum_{t \geq 1} \mathrm{E}[A_t] \leq \mathrm{E}[T] .$$

Thus, since A_t and S_t are independent and $\mathrm{E}[S_t] \leq 1/(2\mathrm{E}[T])$,

$$\mathrm{E}[X] = \mathrm{E}\left[\sum_{t \geq 1} X_t\right] = \mathrm{E}\left[\sum_{t \geq 1} A_t \cdot S_t\right] \leq \frac{1}{2\mathrm{E}[T]} \sum_{t \geq 1} \mathrm{E}[A_t] = \frac{1}{2} .$$

From the Markov inequality it follows that $\Pr[X \geq 1] \leq 1/2$. This means that with probability at least $1/2$ no edge successfully used by P has a latency of more than $2\mathrm{E}[T]$. Hence, the expected runtime for P restricted in a way that it is only allowed to make use of edges with latency at most $2\mathrm{E}[T]$ is at most $2\mathrm{E}[T]$, because otherwise its expected routing time would exceed $\mathrm{E}[T]$. Thus the expected runtime of $\mathcal{S}$ in this case is at most $2\mathrm{E}[T]$, which completes the proof. □

In the following we therefore assume that $\mathcal{S}$ makes use of edges with maximum latency $L \leq 2\mathrm{E}[T]$. Let the random variables C and D be defined such that C denotes the congestion and D denotes the dilation of the path collection chosen by $\mathcal{S}$. Let $B = \max[C, D]$. In order to bound the probability for a large deviation of B from T, we show the following lemma.

Lemma 6.10 *For any* $0 \leq \epsilon \leq 1$ *it holds*

$$\Pr[T \leq (1-\epsilon)B] \leq \mathrm{e}^{-\epsilon^2 B/2L} .$$

Proof. We consider the following two cases.

- $B = C$: Then there exists an edge e that has congestion B. Let k be the number of packets traversing this edge, i.e., the latency of e is $\ell = B/k$. For all $i \in \{1, \dots, k\}$, let the random variable X_i denote the number of time steps it takes for the ith packet to cross e. Then $X = \sum_i X_i$ denotes the time all packets need to traverse e. Since the X_i are independent and geometrically distributed with parameters $p_i = 1/\ell$, Theorem 3.38 yields that, for all $0 \leq \epsilon \leq 1$,

  $$\Pr[X \leq (1-\epsilon)B] \leq \mathrm{e}^{-\epsilon^2 B/2\ell} .$$

  Since $X \leq T$ and $\ell \leq L$, it therefore also holds that

  $$\Pr[T \leq (1-\epsilon)B] \leq \mathrm{e}^{-\epsilon^2 B/2L} .$$

- $B = D$: Then there exists a packet p that traverses a path with dilation B. The runtime for this can be bounded as above by viewing each edge as an independent geometrically distributed random variable. This implies that also in this case,

  $$\Pr[T \leq (1-\epsilon)B] \leq \mathrm{e}^{-\epsilon^2 B/2L} .$$

Combining the two cases yields the lemma. □

Now we can start to bound $\mathrm{E}[B]$. It holds

$$\mathrm{E}[B] \;=\; \sum_{b\geq 1} b\Pr[B=b] \;=\; \sum_{t\geq 1}\Pr[T=t]\sum_{b\geq 1} b\Pr[B=b\mid T=t]\;.$$

Let the function $\epsilon : \mathbb{N} \to \mathbb{R}_+$ be defined as

$$\epsilon(i) = \max\left[\sqrt{\frac{4L(\ln(\mathrm{E}[T]/L)+i)}{t}},\ \frac{4L(\ln(\mathrm{E}[T]/L)+i)}{t}\right]\;.$$

It holds that

$$\Pr[b \geq (1+\epsilon(i))t] \quad\Leftrightarrow\quad \Pr\left[t \leq \left(1-\tfrac{\epsilon(i)}{1+\epsilon(i)}\right)b\right]$$

and, according to Lemma 6.10,

$$\Pr\left[t \leq \left(1-\tfrac{\epsilon(i)}{1+\epsilon(i)}\right)b\right] \leq \mathrm{e}^{-\left(\frac{\epsilon(i)}{1+\epsilon(i)}\right)^2 b/2L} \leq \mathrm{e}^{-\frac{\epsilon(i)^2}{1+\epsilon(i)}\cdot t/2L}\;.$$

If $\epsilon(i)\leq 1$, we have

$$\mathrm{e}^{-\frac{\epsilon(i)^2}{1+\epsilon(i)}\cdot\frac{t}{2L}} \leq \mathrm{e}^{-\frac{4L(\ln(\mathrm{E}[T]/L)+i)}{t}\cdot\frac{t}{4L}} = \frac{L}{\mathrm{E}[T]}\cdot \mathrm{e}^{-i}\;,$$

and if $\epsilon(i) > 1$, we have $\epsilon(i)^2/(1+\epsilon(i)) \geq \epsilon(i)/2$ and therefore

$$\mathrm{e}^{-\frac{\epsilon(i)^2}{1+\epsilon(i)}\cdot\frac{t}{2L}} \leq \mathrm{e}^{-\frac{4L(\ln(\mathrm{E}[T]/L)+i)}{t}\cdot\frac{t}{4L}} = \frac{L}{\mathrm{E}[T]}\cdot \mathrm{e}^{-i}\;.$$

So in any case,

$$\Pr[b\geq(1+\epsilon(i))t] \leq \frac{L}{\mathrm{E}[T]}\cdot\mathrm{e}^{-i}\;.$$

Thus, it follows that

$$\begin{aligned}
\mathrm{E}[B] &= \sum_{t\geq 1}\Pr[T=t]\sum_{b\geq 1} b\Pr[B=b\mid T=t]\\
&= \sum_{t\geq 1}\Pr[T=t]\left[\sum_{b=1}^{(1+\epsilon(1))t} b\Pr[B=b\mid T=t] + \sum_{i\geq 1}(1+\epsilon(i+1))t\cdot\Pr[b\geq(1+\epsilon(i))t]\right]\\
&\leq \sum_{t\geq 1}\Pr[T=t]\left[(1+\epsilon(1))t + \frac{L}{\mathrm{E}[T]}\sum_{i\geq 1}(1+\epsilon(i+1))t\cdot\mathrm{e}^{-i}\right]\\
&= \sum_{t\geq 1}\Pr[T=t]\left[(1+\epsilon(1))t + \frac{L}{\mathrm{E}[T]}\cdot O((1+\epsilon(1))t)\right]\\
&= \sum_{t\geq 1}\Pr[T=t]\left(t+O\left(\sqrt{tL\ln(\mathrm{E}[T]/L)}\right)\right)\left(1+\tfrac{L}{\mathrm{E}[T]}\right)\\
&= \left(1+\tfrac{L}{\mathrm{E}[T]}\right)\left(\mathrm{E}[T]+O\left(\sqrt{L\ln(\mathrm{E}[T]/L)}\right)\mathrm{E}\left[\sqrt{T}\right]\right)\;.
\end{aligned}$$

Let X be an arbitrary random variable. Since $\mathrm{E}[(X-\mathrm{E}[X])^2] = \mathrm{E}[X^2]-\mathrm{E}[X]^2 \geq 0$, it holds $\mathrm{E}[X]^2 \leq \mathrm{E}[X^2]$. Substituting X by $\sqrt{T}$ and taking the root on both sides yields $\mathrm{E}[\sqrt{T}] \leq$

$\sqrt{\mathrm{E}[T]}$. Hence, since $L \leq 2\mathrm{E}[T]$,

$$\begin{aligned}\mathrm{E}[B] &= \left(1+\tfrac{L}{\mathrm{E}[T]}\right)\left(\mathrm{E}[T]+O\left(\sqrt{\mathrm{E}[T]L\ln(\mathrm{E}[T]/L)}\right)\right)\\ &= \mathrm{E}[T]+O\left(\sqrt{\mathrm{E}[T]L\ln(\mathrm{E}[T]/L)}\right)\ .\end{aligned}$$

If we assume that $\mathcal{S}$ is a best possible strategy for routing π, then $\mathrm{E}[T]=R$. (Or $\mathrm{E}[T]\leq 2R$ in case that Lemma 6.9 had to be used. But then $L=\Theta(R)$, which means that $\sqrt{RL\ln(R/L)}=\Theta(R)$.) In this case, $\mathrm{E}[B]=R+O(\sqrt{RL\ln(R/L)})$, which implies that there exists a path collection for routing π with congestion and dilation at most $R+O(\sqrt{RL\ln(R/L)})$. This proves Theorem 6.8. □

Once we have this theorem, the question is whether it is possible to find such a path collection in an efficient way for every permutation routing problem. An option we will consider in this chapter is to construct a *path system*, from which suitable paths for the packets can be easily chosen. A path system is defined as a set of paths in G that contains exactly one path for every pair of nodes in G. The dilation of a path system is defined as for a path collection. The following theorem shows that we can construct efficient path systems. For the rest of the chapter, "w.h.p." means with probability at least $1-N^{-c}$ for any constant c.

Theorem 6.11 *For any PCG G of size N in which every permutation can be routed along a path collection with congestion and dilation at most B and edge latency L, a simple path system $\mathcal{P}$ with dilation at most B can be constructed in polynomial time such that the congestion of routing one packet from every node v to a node chosen uniformly at random by v is bounded by $B+O(\sqrt{(B+L\log N)L\log N})$, w.h.p.*

Proof. First we show the existence of an efficient path system. According to the definition of B, for any permutation π_i on $\{0,\ldots,N-1\}$ with $\pi_i(x)=(x+i) \bmod N$ for all $i,x\in\{0,\ldots,N-1\}$ there is a path collection $\mathcal{P}_i$ with congestion and dilation at most B. We then choose $\mathcal{P}=\bigcup_{i=0}^{N-1}\mathcal{P}_i$ to be the path system for G. Clearly, this path system has congestion at most $N\cdot B$ and dilation at most B.

Next we bound the congestion that holds w.h.p. for routing a random function in G using paths in $\mathcal{P}$. Consider some fixed edge e in G, and let ℓ be its latency. Let the random variable C_e be defined as the congestion at e. For any node v in G, let the binary random variable X_v be one if and only if the packet with source v contains e in its routing path. Then $C_e=\ell\sum_{v\in V}X_v$. Since each node sends out a packet to a destination chosen uniformly at random, we have $\mathrm{E}[C_e]\leq B$. As the X_v are independent, the Hoeffding bounds yield that

$$\Pr[C_e\geq(1+\epsilon)B]\leq\begin{cases}e^{-\epsilon^2B/3\ell} & : \text{ if } 0\leq\epsilon<1\\ e^{-\epsilon\cdot B/3\ell} & : \text{ if } \epsilon\geq 1\end{cases}$$

This probability is polynomially small in N for $\epsilon=O(\max[\sqrt{\frac{\ell\log N}{B}},\frac{\ell\log N}{B}])$ sufficiently large. Hence, w.h.p. all edges have a congestion of at most $B+O(\sqrt{(B+L\log N)L\log N})$. It remains to be shown how to construct an efficient path system.

Constructing an efficient path system

W.l.o.g., we assume that all probabilities attached to the edges in G are of the form $1/\ell$, where $\ell\in\mathbb{N}$. (If this is not the case, then divide all probabilities by N^α, where α is a constant chosen for a sufficiently good precision. Then round all probabilities to the nearest $1/\ell$, where $\ell \in \mathbb{N}$ and continue with the construction as described below. It is easy to see that in this case the relative rounding error per edge is below $1 + N^{-\alpha}$. This ensures that the amount by which the bounds on the congestion and dilation below have to be increased is at most $1 + 1/N^{-\alpha}$, which is insignificantly small.) The following algorithm will serve as a basic building block for our algorithm to construct an efficient path system in G.

For any $d \in \mathbb{N}$, let G_d denote a leveled network of depth d, i.e., a network consisting of $d + 1$ sets of nodes called *levels*. Let its levels be numbered from 0 to d. For each level, let its node set represent the set of all nodes in G. Two nodes v and w are connected if

- v and w are in consecutive levels and represent the same node in G, or
- v and w are in levels i and $i + \ell$ for some $i \geq 0$ and the nodes represented by v and w are connected via an edge with latency ℓ in G.

Let all edges be directed from the lower to the higher level. We consider the problem of sending one unit of flow from every node at level 0 to every node at level d. (That is, we have a so-called multicommodity flow problem.) There are certain capacity limits that have to be kept in order for a solution to our problem to be valid: we demand that for every edge e with latency ℓ in G, all edges in G_d representing copies of e together are allowed to forward a flow of at most $N \cdot d/\ell$. There are no restrictions on how this flow is distributed among these edges. All edges in G_d that connect copies of the same node in G are assumed to have infinite capacity.

If we allow fractional flows, then linear programming can be used to find a solution (or stop with the answer that no solution exists) in polynomial time.

Now we are ready to formulate our algorithm for finding an efficient path system.

(1) Find the minimum d for which there is a solution to the multicommodity flow problem for G_d as stated above.

(2) Transform the multicommodity flow for G_d into a multicommodity flow for G by identifying nodes in G_d representing the same node in G. Since all copies of an edge e in G with latency ℓ together have a capacity of $d \cdot N/\ell$, the fractional congestion of e is at most $d \cdot N$.

(3) Use Raghavan's method [Rag88] for converting fractional flows into integral flows. This results in a path system for G with dilation at most d and congestion at most $d \cdot N + O(\sqrt{(d \cdot N + L \log N) L \log N})$.

Clearly, the algorithm runs in polynomial time. It remains to bound d. According to the existence proof, there is a path system $\mathcal{P}$ with dilation at most B and congestion at most $C = B \cdot N$. This path system can be directly transformed into an integral solution to our multicommodity problem for G_B (the paths of $\mathcal{P}$ represent the paths for the units of flow). Hence, there exists a fractional solution to our multicommodity flow problem with $d \leq B$. Raghavan's rounding method therefore produces a path system with congestion at most $B \cdot N + O(\sqrt{(B \cdot N + L \log N) L \log N})$.

Using the same analysis for bounding the congestion of routing random functions as in the existence proof yields Theorem 6.11. □

Hence, for $L = O(\frac{R}{\log N})$, the congestion and dilation of routing a randomly chosen function using paths in $\mathcal{P}$ is bounded by $O(R)$, w.h.p. Using Valiant's trick [Val82] of routing packets first to randomly chosen intermediate destinations before they are routed to their original destinations, we can get this congestion bound for arbitrary permutations, w.h.p. So if there is an offline protocol that can route packets along an arbitrary simple path collection with congestion C and dilation D in expected time $O(C+D)$, then the *optimal* worst case (and average case, see Claim 6.7) expected time to route packets according to an arbitrary permutation can be reached by using a fixed path system. We will address this question in the next section.

6.2.3 Switching Strategies for PCGs

In this section, we prove upper bounds for the offline and online switching of packets along a fixed collection of paths in a PCG. Recall that in offline switching it is assumed that all nodes know the entire switching problem from the beginning (and are therefore able to compute an optimal schedule since internal computations are usually not counted for the time complexity of switching algorithms), whereas in online switching every node initially only knows the packets it stores.

Offline switching

In order to simplify the development of efficient offline schedules for PCGs, we compare these communication graphs with their deterministic equivalents. For this, we introduce the following type of communication graph.

Definition 6.12 *Let the* deterministic communication graph *(DCG) $G = (V, \gamma)$ be defined as a complete directed graph with node set V and a function $\gamma : V \times V \rightarrow \mathbb{R}_+ \cup \{\infty\}$. For any edge (u, v), $\gamma(u, v)$ represents the time it takes to send a packet from u to v.*

Consider any PCG $G = (V, \varphi)$. Let the DCG of G be defined as $\bar{G} = (V, \gamma)$ with $\gamma(e) = 1/\varphi(e)$ for all $e \in V \times V$. Consider the problem of simulating an offline protocol for $\bar{G}$ by G. The next theorem shows that this can be done efficiently. Its proof can be found in Section 6.4.

Theorem 6.13 *Consider any offline protocol $\mathcal{S}$ that requires T time steps to send all packets along a given path collection of size n in $\bar{G}$. Let L denote the maximum edge latency and ℓ denote the minimum edge latency of this path collection. Then $\mathcal{S}$ can be taken to construct a protocol with runtime $O(T \log(L/\ell) + L \log n)$, w.h.p., for sending the packets along the same path collection in G.*

Note that Theorem 6.13 is true for *any* (even non-simple) path collection. This may allow us to use it for a wider range of problems than just communication problems.

Online switching

With the help of the idea behind the online protocol in [LMR88], we will show the following result.

Theorem 6.14 *There is an online protocol for sending packets along an arbitrary path collection of size n with dilation D, congestion C, maximum edge latency L, and minimum edge latency ℓ in time $O(C + D \log(n \cdot L/\ell))$, w.h.p.*

Proof. We assume in the following that the minimum edge latency, ℓ, is equal to 1. It is easy to modify the proof such that it also holds for an arbitrary minimum edge latency.

Let us first present a protocol for a DCG, called *random delay protocol*, before we show how to convert it to a protocol for a PCG. The protocol assumes that all links have bandwidth B (fixed later), that is, up to B packets can traverse a link at one time step. Let us call an interval of t consecutive time steps a *t-interval*. The following algorithm is used as a basic building block for the random delay protocol.

> **Algorithm Run(s):**
> Each packet is assigned an initial delay, chosen uniformly and independently at random from the range $[C/\log(Ln)]$. A packet that is assigned a delay of δ waits in its initial buffer for δ steps and then moves on without waiting again until it reaches its destination or traversed a path with dilation at least s. If a packet arrives at an edge that is currently traversed by B other packets, then it stops and stays at the link for the rest of Run(s).

The random delay protocol works as follows.

> **repeat**
> execute Run($\min[D, L \cdot n]$)
> **until** all packets reached their destinations

Lemma 6.15 *Suppose we are given an arbitrary simple path collection $\mathcal{P}$ of size n with congestion C, dilation D and edge latency L in some DCG G. Then the random delay protocol needs at most $O(C + D\log(Ln))$ time steps to finish routing in $\mathcal{P}$, w.h.p.*

Proof. Let us consider some fixed edge e and time step t during the execution of Run(s). Let ℓ be the latency of e. Since at most C/ℓ packets want to traverse e and each of these packets chooses an initial delay independently at random from a range of size $C/\log(Ln)$, the probability that at least $B = \max[\alpha + 3, 2\mathrm{e}]\log(Ln) + 3$ packets arrive at e within some ℓ-interval is at most

$$
\begin{aligned}
&(\min[D, L \cdot n] + C/\log(Ln))\binom{C/\ell}{B}\left(\frac{\ell}{C/\log(Ln)}\right)^B \\
\leq\ & 2Ln\left(\frac{\mathrm{e}\log(Ln)}{B}\right)^B \leq 2Ln\left(\frac{1}{2}\right)^{(\alpha+3)\log(Ln)+3} = \frac{1}{4(Ln)^{\alpha+2}}\ .
\end{aligned}
$$

We say that a packet P *fails* at edge e if at that time it reaches it, e is already used by B other packets. Clearly, if P fails then at least B arrivals of other packets fall in an ℓ-interval ending with P's arrival time. Hence, the probability that P fails at least $k = \lceil D/(Ln)\rceil$ times during the execution of the random delay protocol is bounded by

$$
\binom{D+k}{k}\left(\frac{1}{4(Ln)^{\alpha+2}}\right)^k \leq \left(\frac{4D}{k}\right)^k\left(\frac{1}{4(Ln)^{\alpha+2}}\right)^k \leq \left(\frac{1}{(Ln)^{\alpha+1}}\right)^k \leq \frac{1}{(Ln)^{\alpha+1}}\ .
$$

Since there are n packets to consider, the probability that there exists a packet with at least k failures is at most $n \cdot (Ln)^{-\alpha-1} \leq n^{-\alpha}$. Hence, w.h.p. the random delay protocol successfully routes all packets along the given path collection in time

$$
\begin{aligned}
&B \cdot (\min[D, L \cdot n] + C/\log(Ln)) \cdot (D/(\min[D, L \cdot n]) + k) \\
&\quad = \quad O((C + \min[D, L \cdot n] \cdot \log(Ln)) \cdot (D/(\min[D, L \cdot n]) + \lceil D/(Ln)\rceil)) \\
&\quad \overset{C \leq Ln}{=} \quad O(C + D\log(Ln))\ .
\end{aligned}
$$

This completes the proof of Lemma 6.15. □

It remains to be shown how to convert this protocol into a protocol for a PCG. For each time step of the protocol for a DCG with bandwidth B, let us use $4B$ time steps for the PCG.

Consider the simulation of some fixed call of Run(s). Let each packet have a *rank* denoting, for each edge it currently waits at, the time it arrives there in a DCG with bandwidth B. A packet currently staying at an edge with latency ℓ is declared *active* if the current time step t is within the time interval $[4\ell B \cdot r, 4\ell B(r+1))$ of its rank r. Only active packets are allowed to try to cross a link. If several packets are active at the same time at some edge, then the packet with lowest rank is chosen for attempts to cross the edge. (In the case that several packets have the same rank, an arbitrary packet may be chosen.) If a packet is not able to traverse an edge within its active period, it stops and stays at that edge for the rest of Run(s). Also if, for some packet P staying at an edge e of latency ℓ, at least B packets are active at e when P becomes active, P stops and stays at that edge for the rest of Run(s). (Note that every packet that is active at the time when P becomes active has a rank that is at least as low as the rank of P.) If a packet is involved in one of these two cases, we say that it *fails* Run(s).

The rules above imply that every active packet is competing with at most $B-1$ other active packets at any time step during the simulation. Hence, the probability that a packet with rank r fails to traverse its current link e within the time interval $[4\ell B \cdot r, 4\ell B(r+1))$ is at most the probability that e has less than B successes within $4\ell B$ time steps. Let the random variable X denote the number of successes e has within $4\ell B$ time steps. Clearly, $\mathrm{E}[X] = 4B$. Applying the Chernoff bounds with $\epsilon = 1 - \frac{1}{4}$, we obtain

$$\Pr[X \le B] = \Pr[X \le (1-\epsilon)\mathrm{E}[X]] \le \mathrm{e}^{-\epsilon^2 E[X]/2} = \mathrm{e}^{-(1-1/4)^2 4B/2} \le \mathrm{e}^{-B} \ .$$

Using this together with the proof of Lemma 6.15 yields Theorem 6.14. □

With a more involved proof we obtain the following result.

Theorem 6.16 *There is an online protocol for sending packets along an arbitrary simple path collection of size n with dilation D, congestion C, maximum edge latency L and minimum edge latency ℓ in time*

$$O\left(\log(L/\ell)\left(C + D\log\left(L/\ell \cdot \log n\right) + L \cdot \left(\frac{\log(n \cdot L/\ell)}{\log\left(L/\ell \cdot \log n\right)}\right)^2\right)\right) \ ,$$

w.h.p.

Proof. The proof basically combines the ideas in [CMSV96] and [RT96] with Theorem 6.13.

Let us assume in the following that $D \le L \cdot n$. The case $D > L \cdot n$ can be dealt with similar to the proof of Theorem 6.14 (see the way the random delay protocol calls Run(s)). Furthermore, let us assume for the moment that the minimum edge latency, ℓ, is 1. We again first present a protocol for a DCG, called *advanced random delay protocol* (or advanced RDP), before we show how to convert it to a protocol for a PCG. The protocol assumes that all links have bandwidth B (fixed later), that is, up to B packets can traverse a link at one time step. In the following, let

- $p = \frac{\log(Ln)}{\log(L \log n)}$,
- $C_b = b \cdot L\log(Ln) \cdot p$ for some constant $b > 0$ to be set later and

- $D_b = \frac{C_b}{\log(L \log n)}$.

The following algorithm is used as a basic building block for our protocol.

Repeated-Run:
In a first pass, each participating packet chooses an initial delay uniformly and independently at random from the range $[D_b]$. A packet that is assigned a delay of δ waits in its initial buffer for δ steps. Afterwards it moves on in the following way until it reaches its destination or traversed a path with dilation at least D_b:
For each edge e with latency ℓ at which it arrives, it waits for ℓ many steps before traversing e. If, for some edge e with latency ℓ, more than B packets arrive at e during an ℓ-interval, then all of them stop. Packets to which this happens are declared *unsuccessful*.
After the first pass (which takes at most $3D_b$ time steps), $2p$ further passes are performed for the unsuccessful packets. For each pass, every unsuccessful packet chooses an initial delay uniformly and independently at random from the range $[D_b/p]$. The forwarding and stopping of packets in case of a too high contention is done as above. Packets that traversed a path with dilation at least D_b/p stop for this pass. Packets that reach their destination or altogether traversed a path with dilation at least D_b stop for the rest of Repeated-Run.

The advanced RDP then works as follows.

Execute Repeated-Run $2C/C_b + D/D_b$ times. Each packet is assigned an integer, chosen uniformly and independently at random from the range $[2C/C_b]$. A packet that is assigned a value of d awaits d executions of Repeated-Run before participating in (at most) D/D_b consecutive calls of Repeated-Run.

First, let us bound the runtime of the protocol. If all edges have a bandwidth of B and the minimum latency is 1, this is at most

$$\left(\frac{2C}{C_b} + \frac{D}{D_b} + 1\right)(3D_b + 2p \cdot (3D_b/p)) = \left(\frac{2C}{C_b} + \frac{D}{D_b} + 1\right) 9D_b = \frac{18C}{\log(L \log n)} + 9D + 9D_b\ .$$

Hence, for $B = O(\log(L \log n))$ this would result in a runtime of

$$O\left(C + D \log(L \log n) + L \cdot \left(\frac{\log(Ln)}{\log(L \log n)}\right)^2\right)\ .$$

In the case that the minimum latency ℓ is not 1, we scale the parameters C, D, and L to $C' = C/\ell$, $D' = D/\ell$, and $L' = L/\ell$, respectively. In this case we arrive at a time bound of

$$O\left(C' + D' \cdot \log(L' \log n) + L' \cdot \left(\frac{\log(L'n)}{\log(L' \log n)}\right)^2\right)\ .$$

Multiplying this bound with ℓ to obtain the original runtime, we obtain the bound stated in Theorem 6.16. We next show that the probability that a packet fails to reach its destination is very low.

Lemma 6.17 *For any constant $c > 0$ there is a constant $b > 0$ so that for $B = b \log(L \log n)$ and C_b and D_b chosen as above, the probability that the advanced RDP fails to send a packet to its destination is at most n^{-c}.*

Proof. First we bound the probability that the congestion within some call of Repeated-Run exceeds C_b (Claim 6.18). Then we show that under the assumption that the congestion in Repeated-Run is at most C_b, the probability is very small that, for any edge e along the path of a participating packet, there are more than C_b/p packets left with paths containing e that were unsuccessful in the first pass (Claim 6.19). Finally, we show that if the congestion at any edge e caused by unsuccessful packets is at most C_b/p after the first pass, then the probability is very small that one of the unsuccessful packets fails in so many of the $2p$ other passes that it is not able to traverse a path of dilation D_b (Claim 6.20). Combining the three claims yields the lemma.

Claim 6.18 *For any $c > 0$ there is a $b > 0$ such that the probability that there is a call of Repeated-Run with a congestion of more than C_b is at most n^{-c}.*

Proof. Consider some fixed execution of Repeated-Run and some fixed edge e. Let the random variable X denote the congestion caused by the packets that intend to cross e in this call. That is, if e has a latency of ℓ, then each packet that intends to cross e in this call contributes ℓ to X. Since the total congestion at e is at most C and the packets choose at random one of $2C/C_b$ possible starting blocks, $\mathrm{E}[X] \leq C \cdot C_b/(2C) = C_b/2$. Because the packets choose their starting blocks independently at random, we can use the Hoeffding bounds to obtain, for $\epsilon = 1$,

$$\Pr[X \geq C_b] = \Pr\left[X \geq (1+\epsilon)\frac{C_b}{2}\right] \leq \mathrm{e}^{-\epsilon\frac{C_b/2}{3\ell}} \leq \mathrm{e}^{-\frac{b\log^2(Ln)}{6\log(L\log n)}} .$$

Clearly, there are at most $D \cdot n$ different edges and at most $2C/C_b + D/D_b \leq n$ calls of Repeated-Run to consider per edge. Since we assume that $D \leq L \cdot n$, the probability that there is an edge with congestion more than C_b is at most

$$D \cdot n^2 \mathrm{e}^{-\frac{b\log^2(Ln)}{6\log(L\log n)}} \leq (L \cdot n)^3 \cdot \mathrm{e}^{-b\log(Ln)/6} \leq n^{-c}$$

for $b \geq 5(c+3)$. □

Given some fixed call of Repeated-Run and a packet P, a subpath q of P's path is called its *active path segment* if q represents the path P tries to traverse during this call. Then we can show:

Claim 6.19 *Assume that the congestion in any call of Repeated-Run is at most C_b. Then, for any $c > 0$, there is a $b > 0$ such that the probability is at most n^{-c} that, for any edge e along an active path segment, the congestion at e caused by packets that were unsuccessful in the first pass is more than C_b/p.*

Proof. Consider some fixed call of Repeated-Run. A *site* is defined as an ordered pair (e, I), where e is an edge and I is a time interval within the first pass of Repeated-Run. A packet *aims for* a site (e, I) if it selects a random delay such that it intends to arrive at e within I. Furthermore, a packet is said to *participate* in a site (e, I) if it indeed arrives there during I. For any edge e, let I_e denote an interval of length the latency of e. Given a link bandwidth of B and an edge e, a site (e, I_e) is declared *bad* if more than B packets participate in it.

Consider some fixed edge e, and let E_e be defined as the set of all edges contained in the active path segments that cross e. Consider marking any m sites along the edges of E_e. According to our assumptions, e is crossed by at most C_b active path segments and each of these segments contains at most D_b edges. Furthermore, at most $3D_b$ sites have to be considered per edge. Hence, there are at most $C_b D_b \cdot 3D_b$ possibilities of marking a site. Let the random variable X denote the total number of packets aiming for any of the m marked sites. Since the number of active segments containing an edge e' with latency ℓ in E_e is at most C_b/ℓ and each of the corresponding packets chooses a random delay from a range of size D_b, the expected number of packets aiming for any one site of e' is at most $\frac{C_b}{\ell} \cdot \frac{\ell}{D_b} = \log(L \log n)$. Therefore, $\mathrm{E}[X] \leq m \log(L \log n)$. Consider all the marked sites to be bad. Since a packet can only participate in one bad site, we must have $X > B \cdot m$ in this case. As the packets choose their delays independently at random, X can be viewed as the sum of independent binary random variables (each representing one packet). Hence, we can use the Chernoff bounds to bound the probability that X is at least $B \cdot m$ with $B = b \log(L \log n)$ by

$$\Pr[X \geq B \cdot m] \leq \left(\frac{\mathrm{e}^{b-1}}{b^b}\right)^{m \log(L \log n)} .$$

Let $m = \max[\frac{\log(Ln)}{\log(L \log n)}, 3\mathrm{e}]$. Then we get that, for $b = \max[c + 3 \log b + 3, \mathrm{e}^2]$, the probability that there exist m bad sites along the active segments containing e is at most

$$\begin{aligned} \binom{3C_b D_b^2}{m} \left(\frac{\mathrm{e}^{b-1}}{b^b}\right)^{m \log(L \log n)} &\leq \left(\frac{3\mathrm{e}(bL \log^2(Ln))^3}{m}\right)^m \cdot \mathrm{e}^{-bm \log(L \log n)} \\ &\leq \mathrm{e}^{m \cdot 6 \log(\sqrt{bL} \log(Ln)) - bm \log(L \log n)} \\ &\leq \mathrm{e}^{(6 \log \sqrt{b} - b) m \log(L \log n)} \\ &= \mathrm{e}^{-(c+3) \log(Ln)} \leq (Ln)^{-(c+3)} . \end{aligned}$$

This bounds the number of bad sites in E_e and therefore the number of packets that fail and whose active path segments contain e to most $b \log(Ln)$ with probability at least $1 - (Ln)^{-(c+3)}$.

Since there are at most $D_b \cdot n$ edges to consider in a call of Repeated-Run, and there are at most $2C/C_b + D/D_b \leq L \cdot n$ such calls, the probability that some edge has more than $b \log(Ln)$ packets that fail the first pass is at most

$$D_b \cdot L \cdot n^2 (Ln)^{-(c+3)} \leq n^{-c} .$$

Hence, the maximum congestion at any edge after the first pass is at most $Lb \log(Ln) = C_b/p$ with probability at least $1 - n^{-c}$. □

Claim 6.20 *Assume that the congestion at any edge e caused by unsuccessful packets is at most C_b/p after the first pass of any call. Then, for any $c > 0$, there is a $b > 0$ such that the probability is at most n^{-c} that one of the unsuccessful packets fails in so many of the $2p$ other passes that it is not able to traverse a path of dilation D_b in a call of Repeated-Run.*

Proof. Consider some fixed execution of Repeated-Run. Within this execution, consider some fixed packet P and edge e in P's active path segment. Let ℓ be the latency of e. The probability that P runs at e into a bad site is at most

$$\binom{C_b/(\ell \cdot p)}{B} \left(\frac{\ell}{D_b/b}\right)^B \leq \left(\frac{\mathrm{e}}{b}\right)^B .$$

Let $B = b \log(L \log n)$ and $b \geq 4\mathrm{e}$. Then the probability that P fails a pass is at most

$$\frac{D_b}{p} \cdot \left(\frac{\mathrm{e}}{b}\right)^B \leq bL \log^2(Ln) \cdot \left(\frac{1}{2}\right)^{2b \log(L \log n)} \leq (L \log n)^{-(b+1)} .$$

If P fails the call of Repeated-Run, then it must have failed in more than p passes. The probability for this is at most

$$\binom{2p}{p}\left(\frac{1}{L\log n}\right)^{(b+1)p} \leq \left(\frac{1}{L\log n}\right)^{bp} \leq n^{-b} .$$

Summing up over all possible packets and calls of Repeated-Run yields the claim for $c \leq b-2$. □

The claims complete the proof of Lemma 6.17. □

It remains to be shown how to convert this protocol into a protocol for a PCG. Our aim will be to use the proof of Theorem 6.13. For this we modify our protocol in the following way:

Consider some fixed pass of a call of Repeated-Run. Every packet participating in this pass still chooses a random initial delay, but afterwards it tries to traverse its path *without* waiting a certain amount of time at each edge. (The waiting in the protocol for the DCG was only necessary to ensure that all packets participating in a bad site can be stopped.) To be able to use the results of Theorem 6.13, we apply the same sequence of transformations to this protocol as described in Section 6.4.2. Since these transformations preserve the order in which the packets pass through the edges, the probability bounds above for more than B packets to intend to use the same edge at the same time still hold. Let schedule $\mathcal{S}$ be the result of these transformations. For the simulation on a PCG, let each packet have a *rank* denoting, for each edge it currently waits at, the difference between the actual time step and B times the time step at which it is supposed to cross this edge in the DCG $\bar{G}$ with bandwidth B, using $\mathcal{S}$. Since $\mathcal{S}$ ensures that the traversal of any edge of latency ℓ always starts at an integral multiple of ℓ, we use the following vital stopping rule for the simulation: if, at any time, there are more than B packets with the same rank waiting to traverse the same edge in the PCG, then all of them stop for the rest of the simulation of $\mathcal{S}$.

To be able to apply the analysis for the advanced RDP to $\mathcal{S}$, the definition of a bad site has to be slightly changed. Now, a site is called bad if more than B packets with the same rank wait for transmission at the same time at some edge e, instead of intending to use it within a certain time interval corresponding to the latency of e. This new definition still ensures that every packet can only be used once as a witness of a bad site, since it is halted at the corresponding edge by the rule above. Since each packet selects its rank independently at random, the independence assumptions in Claim 6.19 are therefore still correct, which implies that Lemma 6.17 and thus the runtime bound shown above are still correct. Furthermore, our new stopping rule ensures the following fact for all packets to which the stopping rule does *not* apply, replacing Fact 6.39 in the proof of Theorem 6.13.

Fact 6.21 *A packet can only be blocked from transmission attempts by at most $B-1$ packets of equal rank or a packet of strictly higher rank. Furthermore, for any given rank r, any packet of rank more than r can block at most B packets of rank r.*

As we will see in Section 6.4, this fact ensures that we can treat our online protocol as a *legal* offline protocol (i.e., a protocol without transmission conflicts) for a DCG of bandwidth B. This offline protocol is to be simulated on a PCG G with link bandwidth 1. An easy modification of the proof of Theorem 6.13 shows that this can be done in $O(B \cdot T \log(L/\ell) + L\log n)$ time steps, w.h.p., which concludes the proof of Theorem 6.16. □

6.2.4 Putting Everything Together

Theorem 6.6 together with Claim 6.7, Theorem 6.11 and Theorem 6.14 show that for *any* transmission graph using a scheme out of our class of local probabilistic control schemes there is an online protocol that can reach up to a factor of $O(\log N)$ for arbitrary permutations the optimal average case permutation routing time. Theorem 6.16 further shows that this can be reduced to a factor of $O((\log \log N)^2)$ if $L = \log^{O(1)} N$ and $L \leq \frac{R}{\log^2 N}$. It would be interesting to see, whether further improvements are possible.

6.3 Wireless Communication in a Euclidean Space

We next turn to the case where the mobile stations that wish to communicate with each other are points in $\mathbb{R}^d$, and the weight of an edge that connects any two points p_1 and p_2 is the Euclidean distance between the two points in $\mathbb{R}^d$. This models the scenario where the required signal strength to send a message between a pair of nodes is only a function of distance. The cases where $d = 2$ and $d = 3$ are the most relevant from a practical standpoint. Here we focus on the case where $d = 2$. However, all of our results can be easily extended to apply also to the case where $d = 3$. Recall that we assume the nodes to be distributed independently and uniformly at random in a fixed region of $\mathbb{R}^2$, called the domain space.

6.3.1 Upper Bounds

Square domain spaces

We start with the case where the domain space is exactly a square. We demonstrate how to take advantage of the considerable similarity between randomly distributed nodes in such a domain space and existing work on computing with faulty arrays [Rag89, KKL$^+$90, CMS97]. For $n = a^2$ for some integer a, let an *n-partition* of a square domain space be the partition of the domain space into exactly n equal sized square regions and let r_{ij} denote the region in the ith row and the jth column, counting from some fixed corner of the domain space. The following easy lemma is useful in many of our later proofs.

Lemma 6.22 *For any n-partition and any communication pattern where the nodes in every region send at most h messages and every message sent from region r_{ij} has a destination node in region r_{ij}, or in a region that abuts region r_{ij}, there exists a schedule that sends every message in time $O(h)$.*

Proof. Recall that α, the ratio of the interference radius to the transmission radius, is a constant. Regardless of where in a region the nodes are placed, every message sent from region r_{ij} directly to its destination conflicts only with regions $r_{i'j'}$ with $|i - i'|$ and $|j - j'|$ at most $d = \lceil 3\alpha \rceil$. Thus, for any $0 \leq x, y < d$, all messages sent from any region r_{ij} where $i = x \bmod d$ and $j = y \bmod d$ can be sent concurrently. Hence, all messages can be sent in d^2 waves. Since each wave requires time $O(h)$, the lemma follows. □

This implies, for example, that if we have an n-partition with exactly one communication node located somewhere in each region, then in a constant number of steps we can route any communication pattern performed in a single step on a (non-faulty) $\sqrt{n} \times \sqrt{n}$ array. Here, the node in region r_{ij} performs the communication performed by processor p_{ij} of the array.

When each node is distributed uniformly and independently at random, it is very unlikely that there will be exactly one node in each region of the n-partition. Let $S_{n,m}$ be a placement of n nodes into an m-partition. Given any $S_{n,m}$, let the $\sqrt{m} \times \sqrt{m}$ faulty array $F(S_{n,m})$ be defined such that processor p_{ij} of $F(S_{n,m})$ is fault-free if $S_{n,m}$ has at least one node in region r_{ij} of the m-partition, and is faulty otherwise. A faulty processor cannot send or receive any messages. Due to Lemma 6.22, $S_{n,m}$ can emulate $F(S_{n,m})$ with constant slowdown. Note that in most cases the number of nodes in $S_{n,m}$ is strictly larger than the number of fault-free processors in $F(S_{n,m})$, since some of the regions of the domain space contain more than one node. When region r_{ij} contains more than 1 node, one arbitrarily chosen node in the region performs the communication performed by processor p_{ij} of the array.

Existing work on routing with faulty arrays considers the scenario where each node fails independently with probability p [Rag89, KKL$^+$90, CMS97]. Thus, in order to simulate algorithms for the faulty array, we need to deal with the fact that the event that a given square contains some node is dependent on how many other squares also contain nodes. We do this by requiring that the algorithm for the array to be simulated is (p, k, m)-*guaranteed*, a technical condition described below. This technical condition is actually fairly natural, and most existing algorithms for faulty arrays adhere to it.

In order to describe an algorithm that is (p, k, m)-*guaranteed*, we first need to define some terms. Let S denote any subset of the processors of the faulty array A, and let $f_A(S)$ denote the set of fault-free processors of A in S. An *array property* specifies a subset S and a function $P(f_A(S))$ from $f_A(S)$ to $\{0, 1\}$. If it is clear from the context to which array f is applied, we will write $f(S)$ instead of $f_A(S)$.

Definition 6.23 *A* monotonic array property *is an array property $P(f_A(S))$ that ensures that for any two faulty arrays A and A', where $f_A(S) \subset f_{A'}(S)$, if $P(f_A(S)) = 1$, then $P(f_{A'}(S)) = 1$.*

Most algorithms for faulty arrays work correctly provided that a subset of the processors with a specified structure is fault-free: such a requirement is a monotonic array property. An example of an algorithm that might not be so guaranteed is one where the question of whether a processor is faulty or not is used to provide random bits.

Definition 6.24 *A set of k array properties $\{P_1(f(A_1)), \ldots, P_k(f(A_k))\}$ is said to (p, k, m)-* guarantee *an algorithm for the faulty array A if for all $1 \le j \le k$, $\Pr[P_j(f(A_j)) = 0] \le p$, the number of processors in A_j is at most m, and the algorithm functions correctly whenever $P_j(f(A_j)) = 1$ for all $1 \le j \le k$.*

In other words, an algorithm is (p, k, m)-guaranteed if we can specify a set of k (not necessarily disjoint) subsets of the processors, each of size at most m, and each with an array property that is 1 with probability at least $1 - p$, such that if all the array properties are satisfied then the algorithm functions correctly.

Lemma 6.25 *Let A be a $\sqrt{n} \times \sqrt{n}$ faulty array where every processor is faulty independently with probability q, for any constant q. Let P_j be any monotonic array property on A_j, a subset of at most $\frac{n}{2}$ of the processors of A. There exists a constant c such that for the random placement $S_{cn,n}$ of cn wireless nodes in an n-partition,*

$$\Pr[P_j(f(F(S_{cn,n})_j)) = 1] \ge \Pr[P_j(f(A_j)) = 1] \ ,$$

where $F(S_{cn,n})_j$ denotes the set of processors in $F(S_{cn,n})$ corresponding to A_j in A.

Proof. For some fixed ordering of the processors in A_j, let the binary random variable X_i, $1 \leq i \leq |A_j|$, be equal to 1 if processor i of A_j is fault-free and 0 otherwise. Furthermore, let the binary random variable Y_i, $1 \leq i \leq |A_j|$, be 1 if processor i of $F(S_{cn,n})_j$ is fault-free and 0 otherwise.

Claim 6.26 *There exists a constant $c > 0$ such that for Y_i defined by $S_{cn,n}$,*

$$\Pr[Y_i = 1 \mid Y_1 \dots Y_{i-1}, Y_{i+1} \dots Y_{|A_j|}] \geq q \; .$$

Proof. We prove the claim in two steps. First, we determine the total number of fault-free processors in $F(S_{cn,n})$, and then we prove the probability bound for Y_i conditioned on this number.

For some fixed ordering of all processors in $F(S_{cn,n})$, let the binary random variable Z_i, $1 \leq i \leq n$, be 1 if processor i of $F(S_{cn,n})$ is fault-free and 0 otherwise. Furthermore, let $N = \sum_i Z_i$ represent the total number of fault-free nodes in $F(S_{cn,n})$. Obviously,

$$\mathrm{E}[N] = n \cdot (1 - (1 - \tfrac{1}{n})^{c \cdot n}) \geq n(1 - \mathrm{e}^{-c}) \; .$$

Furthermore, using Theorem 3.59, when c is a constant, it holds that

$$\Pr[N < (1 - 2\mathrm{e}^{-c})n] \leq \mathrm{e}^{-\Omega(n)} \; .$$

We first condition on the value of N. If we then condition on the values of $Y_1 \dots Y_{i-1}, Y_{i+1} \dots Y_{|A_j|}$, the setting of these random variables that minimizes $\Pr[Y_i = 1]$ is when $Y_1 = Y_2 = \dots Y_{i-1} = Y_{i+1} = \dots Y_{|A_j|} = 1$. Thus,

$$\begin{aligned} \Pr[Y_i = 1 \mid N, Y_1 \dots Y_{i-1}, Y_{i+1} \dots Y_{|A_j|}] &\geq \frac{N - |A_j|}{n - (|A_j| - 1)} \geq \frac{N - n/2}{n + 1 - n/2} \\ &= \frac{2N - n}{n + 2} \; . \end{aligned}$$

Therefore, by Bayes's rule we have

$$\begin{aligned} &\Pr[Y_i = 1 \mid Y_1 \dots Y_{i-1}, Y_{i+1} \dots Y_{|A_j|}] \\ &\quad \geq \Pr[N \geq (1 - 2\mathrm{e}^{-c})n] \cdot \Pr[Y_i = 1 \mid N, Y_1 \dots Y_{i-1}, Y_{i+1} \dots Y_{|A_j|}] \\ &\quad \geq (1 - \mathrm{e}^{-\Omega(n)}) \cdot \frac{2(1 - 2\mathrm{e}^{-c})n - n}{n} \; . \end{aligned} \tag{6.1}$$

Obviously, for any constant $0 < q < 1$, there is a constant c such that (6.1) $> q$. □

Given this claim, we prove the lemma as follows. Deciding which $X_i = 1$ (i.e., deciding which processors of the faulty array A_j are fault-free) can be viewed as traversing a complete binary tree T of depth $|A_j|$ from the root to a leaf, where at level i, we branch right if $X_i = 1$ and left otherwise. Let $A_j(x)$ denote the configuration of faulty and non-faulty processors in A_j represented by the xth leaf of T (counted in the left to right ordering of the leaves of T). For some of the leaves of T, $P_j(f(A_j(x))) = 1$. We label every node v of T with a probability p_v, which is the probability of reaching any leaf $A_j(x)$ such that $P_j(f(A_j(x))) = 1$, given that we have already reached node v. We label both outgoing edges from node v with the probability of taking that edge from v. In T, all right branches have label q, and all left branches have label $1 - q$.

For $S_{cn,n}$, we define an analogous tree T', where at level i of T' we branch right if $Y_i = 1$, and we branch left otherwise. We label every node v' of T' with a probability $p_{v'}$, which is the probability of reaching a leaf $F(S_{cn,n})_j(x)$ such that $P_j(f(F(S_{cn,n})_j(x))) = 1$, given that we have already reached node v'. We label both outgoing edges from node v' with the probability of taking that edge from v'. If node r is the root of T and node r' is the root of T', then to prove the lemma, we need to show that $p_r \leq p_{r'}$.

To do this, we use a series of trees $T_1 \ldots T_{|A_j|}$, where $T_1 = T$ and $T_{|A_j|} = T'$. T_j is the complete binary tree where all edge labels between nodes which both have height at most j (counting the root here as height 1) are identical to the corresponding labels in the tree T', and the remainder of the edge labels are identical to the corresponding labels in the tree T. We show that for every node in T_j, the node label of the corresponding node in T_{j+1} is at least as large, and the lemma follows directly by induction.

We use two facts. For any internal node v of T or T', let $l(v)$ and $r(v)$ be the left and right children of v respectively. Fact 1 is that for any node v' of T', the probability of proceeding from v' to $r(v')$ is at least q. This follows from Claim 6.26. Fact 2 is that for any node v of T, $p_{r(v)} \geq p_{l(v)}$. This follows from the fact $P_j(A_j)$ is monotonic and the fact that edge labels are identical in the subtrees rooted at $p_{r(v)}$ and $p_{l(v)}$.

Note that in T_j all node labels at levels greater than j are the same as their value in T. Thus, by Fact 2, every node w in level j of T_j is such that $p_{r(w)} \geq p_{l(w)}$. Furthermore, by Fact 1, when the edge labels between levels j and $j+1$ are changed from their values in T_j to their values in T_{j+1}, this cannot decrease the likelihood of going from w to $r(w)$. Thus, all node labels at level j of T_{j+1} are at least as large as the corresponding node label in T_j. It is easy to show by induction that this implies that all node labels of T_{j+1} are at least as large as the corresponding node label in T_j. □

This lemma and Lemma 6.22 give us the following result.

Theorem 6.27 *Let $\mathcal{A}$ be an algorithm for A, a $\sqrt{n} \times \sqrt{n}$ faulty array with each processor being faulty independently with probability q for some constant q, where $\mathcal{A}$ is $(p, k, \frac{n}{2})$-guaranteed by a set of monotonic array properties. With probability at least $1 - kp$, a random placement of $O(n)$ wireless nodes into a square domain space can simulate algorithm $\mathcal{A}$ with constant factor slowdown per step of $\mathcal{A}$.*

So, for example, we can use the results of [KKL+90] to obtain the following:

Theorem 6.28 *With probability $1 - O(1/n)$, a placement of n wireless nodes uniformly and independently at random into the domain space can perform any of the following in time $O(\sqrt{n})$:*

- *Route an arbitrary online permutation between the n nodes.*
- *Sort n keys.*
- *Multiply two $\sqrt{n} \times \sqrt{n}$ matrices.*

The algorithms for all of these are deterministic and require constant sized queues.

Proof. [KKL+90] provides algorithms that run in time $O(\sqrt{n})$ for all of these problems for the $\sqrt{n} \times \sqrt{n}$ faulty array. They show that these algorithms work correctly, provided that the array is $\sqrt{n}/4$-*gridlike*. An array is r-gridlike if every $r \times r$ subarray has at least $\frac{2}{3}r$ fault-free paths connecting the left and right sides of the subarray and at least $\frac{2}{3}r$ fault-free paths connecting the top and bottom of the subarray. Being r-gridlike is a monotonic array property and is also decomposable: if the processors in columns 0 through $\frac{\sqrt{n}}{2}$ are r-gridlike, the processors in columns $\frac{\sqrt{n}}{4}$ through $\frac{3\sqrt{n}}{4}$ are r-gridlike, and the processors in columns $\frac{\sqrt{n}}{2}$ through $\sqrt{n}$ are r-gridlike, then the entire array is r-gridlike, provided that $r \leq \frac{\sqrt{n}}{4}$. In [KKL$^+$90] also the following result is shown.

Lemma 6.29 ([KKL$^+$90]) *A $\sqrt{n} \times \sqrt{n}$ array, where each processor is faulty with probability p, is $\frac{logn}{\log 1/p}$-gridlike with probability at least $1 - \frac{1}{n}$.*

Thus the algorithms for the three problems are $(\frac{1}{n}, 3, \frac{n}{2})$-guaranteed by a set of monotonic array properties. For the routing result, the algorithm of [KKL$^+$90] is only able to route permutations consisting of pairs of processors connected by a path of fault-free processors. However, we can use the extra power of wireless communication to route *any* permutation between all n nodes. We do this by using the $\frac{n}{\log^2 n}$-partition of the domain space. Call each square of the $\frac{n}{\log^2 n}$-partition a super-region. Using the Chernoff bounds, we can show that every super-region has $O(\log^2 n)$ nodes w.h.p. Furthermore, it is shown in [KKL$^+$90] that in any $\log n \times \log n$ region of the faulty array, there are $\Omega(\log^2 n)$ *active* processors, where each active processor has a fault-free path to every other active processor in the faulty array. Thus, Lemma 6.22 implies that in time $O(\log^2 n)$ we can redistribute the messages within each super-region so that every message is at an active node, and each active node has $O(1)$ messages, completing the proof. □

Arbitrary convex regions

We also demonstrate that these techniques can be extended to route a random permutation in a time that is asymptotically optimal when the nodes are distributed independently and uniformly at random in any 2-dimensional convex region. In such a domain space, an n-partition is defined as follows. Find the longest chord of the domain space, called the major axis, and the longest chord perpendicular to the major axis, called the minor axis. The n-partition is the set of n abutting squares with largest side length that fit *entirely* into the domain space, where the squares are aligned with the major and minor axes. The *width* of the domain space is defined to be the number of squares in the n-partition along the major axis, and the *height* is the number along the minor axis. Note that the height of a domain space is always at most its width. As a first step, we consider the problem of nodes distributed randomly in any rectangular region.

Lemma 6.30 *For the case that n nodes are distributed independently and uniformly at random in a rectangular domain space with height H and width $\frac{n}{H}$, a random permutation can be routed in time $O(\frac{n}{H} \cdot \min[H, \lceil\frac{\log n}{H}\rceil])$, w.h.p.*

Proof. We use the n-partition of the rectangular domain space, and the corresponding faulty array. When $H \geq \log n$, the routing algorithm follows directly from the techniques of the previous subsection, since the rectangle can be shown to be $\log n$-gridlike. Also, when $H \leq \sqrt{\log n}$, each message can be sent directly from its source to its destination in the stated running time. Thus, the interesting case is when $\sqrt{\log n} < H < \log n$.

In this case, we use the $\frac{nH^2}{\log^2 n}$-partition of the rectangular domain space. (We assume here w.l.o.g. that $\frac{\sqrt{n}H}{\log n}$ is an integer.) Call each square of this partition a *super-region*. Each of these super-regions consists of a square of $\frac{\log n}{H} \times \frac{\log n}{H}$ regions of the n-partition. Let S' be the distribution of the nodes in the $\frac{nH^2}{\log^2 n}$-partition, and let $F(S')$ be the corresponding faulty $\frac{n}{\log n} \times \frac{H^2}{\log n}$ array, where a processor of $F(S')$ is fault-free if and only if the corresponding super-region of S' contains at least one node. When each region of the n-partition has a node in it independently with probability $\frac{1}{3}$, then each super-region has at least one node in it with probability $1 - p$, where $p = (\frac{2}{3})^{\frac{\log^2 n}{H^2}}$. Let A' be the faulty $\frac{n}{\log n} \times \frac{H^2}{\log n}$ array, where each processor is faulty independently with probability p.

By Lemma 6.29, array A' is $\frac{\log n}{\log 1/p} = \Theta(\frac{H^2}{\log n})$-gridlike, w.h.p. Thus, A' can route a one-item per fault-free processor routing problem in time $O(\frac{n}{\log n})$. By Lemma 6.25, the array $F(S')$ is also $\Theta(\frac{H^2}{\log n})$-gridlike, and thus by Lemma 6.22, the nodes in S' can also perform, in time $O(\frac{n}{\log n})$, any routing problem where there is at most one packet for each super-region that contains at least one node. However, the distribution of wireless nodes actually has an average of $\frac{\log^2 n}{H^2}$ packets per super-region. Since every set of $\frac{H^2}{\log n}$ super-regions has $O(\log n)$ packets w.h.p., the packets can be redistributed in time $O(\log n)$ so that in fact each super-region has $O(\frac{\log^2 n}{H^2})$ packets. Thus, by solving $O(\frac{\log^2 n}{H^2})$ separate routing problems, the total time required is $O(\frac{n \log n}{H^2})$. □

We now turn to the case of domain spaces that are not rectangles.

Theorem 6.31 *For n nodes distributed uniformly and independently at random into a convex domain space with height H, there is an algorithm that, with probability $1 - O(1/n)$, routes a random permutation in time $O(\frac{n}{H} \min[H, \lceil \frac{\log n}{H} \rceil])$.*

Proof. We use an n-partition of the convex domain space. Recall that we require all regions of the n-partition to be completely inside the domain space. It is not difficult to verify that at most an $O(1/H)$ fraction of the area covered by the domain space is not contained in any region of the n-partition. Since w.h.p. there are at most $O(n/H)$ nodes that are placed in this uncovered area, all messages that start or end at any of these nodes can be routed in time $O(n/H)$. Thus it remains to find a strategy for those messages that have their start and end point in the n-partition.

If $H \leq \sqrt{\log n}$ then our upper bound is trivially true. When $H \geq \log^2 n$, we use super-regions consisting of $\log n \times \log n$ squares that are completely inside the domain space. The only regions of the n-partition that are more than $\log n$ regions away from any super-region are the regions at the extreme end-points of the major axis. Let us call these regions *distant regions*. By the convexity assumption, the domain space can have a height of less than $\log n$ for a length of at most $O(\frac{n/H}{H/\log n}) = O(\frac{n \log n}{H^2})$ along its major axis. Thus, the number of distant regions is at most $O(\log n \cdot \frac{n \log n}{H^2}) = O(\frac{n \log^2 n}{H^2})$, and therefore routing directly to and from the nodes in those regions can be performed in time $O(\frac{n}{H})$. For the remainder of the nodes, we route to the nearest node in a super-region by routing either first to the correct row and then to the correct column, or vice versa (depending on which path does not leave the domain space). Then, from there, the packet is forwarded to the correct node.

It remains to consider the situation that $\sqrt{\log n} < H < \log^2 n$. In this case, the number of nodes that are not within a distance of $\log n$ of a super-region can be so high that we need to be more careful. We partition the domain space as follows: Where the local height of the space is $> \log n$, we partition the space into the same $\log n \times \log n$ super-regions as before. When the local height is $< \log n$, we partition the domain space into rectangles of height 2^i with $i \in \{1, \ldots, \log \log n\}$. By the convexity assumption, there are at most two portions of the domain space where the local height is between 2^i and 2^{i+1}, for any integer $i \leq \log \log n$. Each of these portions is given to a rectangle of height 2^i. Note that if we examine the arrangement of rectangles along the major axis, then from both sides of the domain space this results in rectangles of increasing size until we possibly reach a portion of the domain space of height $> \log n$.

We now describe how to route packets that travel in the direction of one endpoint of the major axis (called without loss of generality the left endpoint of the domain space). We assume that the packets start at uniformly distributed nodes in the area of largest height in the domain space. With the ability to manage this, we can route any permutation of packets, since routing in the opposite direction is just the reverse process.

We number the rectangles between the left endpoint and the point of largest height in the domain space from smallest to largest: the jth smallest rectangle is R_j, and let R_m, $m = \min[\log H, \log\log n]$ be the largest such rectangle. Suppose that the width of some rectangle R_j is $o(\log n)$. Then, by the convexity assumption, the number of nodes in the rectangles R_i, $i \leq j$, is at most $o(\log^2 n)$, which means that any packets destined for these nodes can be delivered directly to their destinations in the stated time. Thus, we consider in the following only the problem of routing along rectangles of width $\Omega(\log n)$.

Next we show that we can further reduce our problem to considering only those rectangles R_j with $j \geq \frac{1}{2}\log\log n$. By the convexity argument, R_j can have a width of at most $O(\frac{n/H}{H/2^j}) = O(\frac{2^j n}{H^2})$. Thus, the number of regions in R_j is at most $O(2^j \cdot \frac{2^j n}{H^2})$. This implies that the total number of regions in rectangles R_k, $k \leq j$, is at most $O(\frac{2^{2j}n}{H^2})$. W.h.p., the number of packets bound for these rectangles is also $O(\frac{2^{2j}n}{H^2})$. For $j \leq \frac{1}{2}\log\log n$ this number is at most $O(\frac{n\log n}{H^2})$. Thus we can send all packets destined for rectangles R_j with $j \leq \frac{1}{2}\log\log n$ directly to their destinations in the stated time.

Hence, it remains to consider those rectangles R_j with $\frac{1}{2}\log\log n \leq j \leq m$. By using our results for rectangular domain spaces, each R_j can be shown to be $\Theta(\frac{2^{2j}}{\log n})$-gridlike. Thus, there are $\Theta(\frac{2^{2j}}{\log n})$ paths in rectangle R_j from the left side to the right side (including some interface area of R_{j+1}). To simplify the following arguments, we assume that there are exactly $\frac{2^{2j}}{\log n}$ such paths. In this case, there are exactly 4 times as many paths in R_{j+1} as in R_j. These paths can be connected (using paths from the upper side to the lower side of R_{j+1}) so that every path in R_j is connected to 4 paths in R_{j+1}, every path in R_{j+1} is connected to a path in R_j, and no two paths in R_{j+1} connected to different paths in R_j intersect (see Figure 6.1).

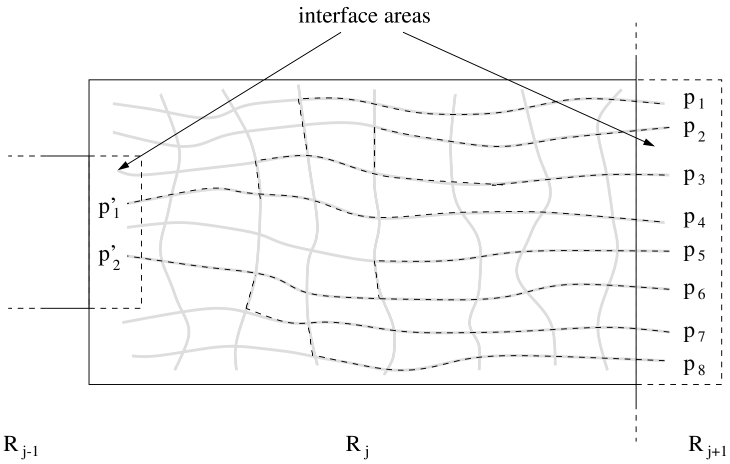

Figure 6.1: Connecting the paths in the rectangles.

With such a setup of paths, we route as follows. Each packet destined to some rectangle R_j selects one of the $\frac{2^{2m}}{\log n}$ paths of R_m uniformly and independently at random. The packets are first sent to the starting points of their selected paths in (the interface area of) rectangle R_m, using the previous faulty-array-like routing techniques. Afterwards, each packet is routed along the chosen path to the correct column of the domain space. Since a path may be used by more than one packet, there may be conflicts between two packets trying to use the same path. When there is a conflict between two packets, the packet that has the furthest to go along the path is sent first. After all the packets have reached the correct columns, the packets are sent directly to their destinations.

It is easily shown that the total number of times a packet is transmitted is at most $O(\frac{n}{H})$. Thus, it only remains to show that the packets are delayed by conflicts for the same path for at most $O(\frac{n \log n}{H^2})$ steps. Consider a packet that is destined for rectangle R_j. It can only be delayed by packets on their way to rectangles R_k with $k \leq j$. As argued above, w.h.p., the number of packets bound for these rectangles is at most $O(\frac{2^{2j} n}{H^2})$. Since each of these packets passes along a randomly chosen path, the number of these packets traveling along each path is w.h.p. $O(\frac{2^{2j} n}{H^2} \cdot \frac{\log n}{2^{2j}}) = O(\frac{n \log n}{H^2})$. This means that any packet destined for R_j can be delayed for at most $O(\frac{n \log n}{H^2})$ steps, which proves the theorem. □

6.3.2 Lower Bounds

In this section we address the question of how much additional power is provided by the ability to be able to communicate directly with nodes that are far away. The wireless scenario might be faster than an array, since a wireless transmission can send any single message from its source to its destination in a single step. The cost of this, however, is that such a transmission can block many other transmissions from occurring. We demonstrate that in the Euclidean case, the use of such long transmissions does not help for most permutations, and in fact the results of the previous section are within a constant factor of optimal. We assume that packets to be transmitted are indivisible (i.e., that the algorithm does not encode or duplicate the content of the packets), and each transmission by a node can forward at most one packet.

Theorem 6.32 *For n nodes placed uniformly and independently at random in a square domain space, the time to route a random permutation is w.h.p. $\Omega(\sqrt{n})$.*

Proof. In the following, let (xy) denote the distance from node x to node y, where the unit of distance is defined to be the side length of a square of the n-partition. In this proof, in some cases, instead of using the distance between two points, we use the distance between the centers of the respective regions of the n-partition. Thus, we start with a fact that bounds the difference in distance between these two measures of distance.

Fact 6.33 *For $d > \sqrt{2}$, let $\beta_d = 1 - \frac{\sqrt{2}}{d}$. Let x and y be any two points that are at least a distance of d apart, and let $\bar{x}$ be the exact center of the region containing x. Then*

$$(xy) \geq \beta_d (\bar{x}y)$$

and

$$(\bar{x}y) \geq \beta_d (xy)$$

In the following, let k be the minimum integer such that $\frac{1}{\beta_k} \leq \sqrt[3]{\alpha}$ and $(\alpha - 1)k > \sqrt{5}$. Note that since α is a constant > 1, k is also a constant. For this k, we see that if node a transmits to node a' on a step where node b' also receives a message, where $(aa') \geq k$ and $(ab') \geq k$, then

$$(\bar{a}b') \geq \alpha^{-1/3}(ab') \geq \alpha^{2/3}(aa') \geq \alpha^{1/3}(\bar{a}a') \ . \tag{6.2}$$

Furthermore, this k ensures that if two nodes are in the same region or in abutting regions, then only one of them can transmit a message over a distance of at least k at the same time, since for $(\alpha - 1)k > \sqrt{5}$ the difference between the interference radius and the transmission radius is larger than the distance of these two nodes.

Definition 6.34 *We say that region r_{ij} is k-*occupied *at time t by message P if message P is at a node in region r_{il}, $l + k < j$, at time $t - 1$, and message P is at a node in region r_{xy}, $y \geq j$, at time t.*

Thus, every time a message moves r regions to the right, $\max[r - k, 0]$ regions are k-occupied. This holds regardless of how many columns a packet moves during any step, and also regardless of how many rows the packet moves up or down during any step. Let o be the total number of regions that are k-occupied by messages over the course of the routing algorithm. Let p be the total number of transmissions that move a packet somewhere between 1 and k regions to the right. In a random permutation, the total, over all packets, of the number of columns moved from left to right is w.h.p. $\Omega(n\sqrt{n})$. However, when a message moves ℓ columns to the right, if $\ell \leq k$, the contribution of this step to p is 1, and if $\ell > k$, the contribution of this step to o is at least $\ell - k$. Thus, each column moved to the right by any message contributes at least $\frac{1}{k+1}$ to the sum $o + p$, Hence, in a random permutation, $o + p = \Omega(n^{3/2})$. This means that $\max[p, o] = \Omega(n^{3/2})$. If $p = \Omega(n^{3/2})$, the theorem follows directly from the fact that at most n messages can be sent at any step. Otherwise, the following lemma suffices.

Lemma 6.35 *For any time step t and for any region r, at most $O(1)$ messages can k-occupy r at step t.*

Proof. We assume that at some time step, nodes $a_1, a_2, \ldots a_j$ each successfully transmit to nodes $a'_1, a'_2, \ldots a'_j$ respectively, and they all k-occupy some region s. We shall see that j is bounded by a constant. The nodes are ordered so that a_1 is the closest to (the center of) s and a_j is the furthest. According to our choice of k above, the regions of the a_i's must be at least one region apart. Let $\bar{a}_i$ be the midpoint of the region containing a_i. This is depicted in Figure 6.2.

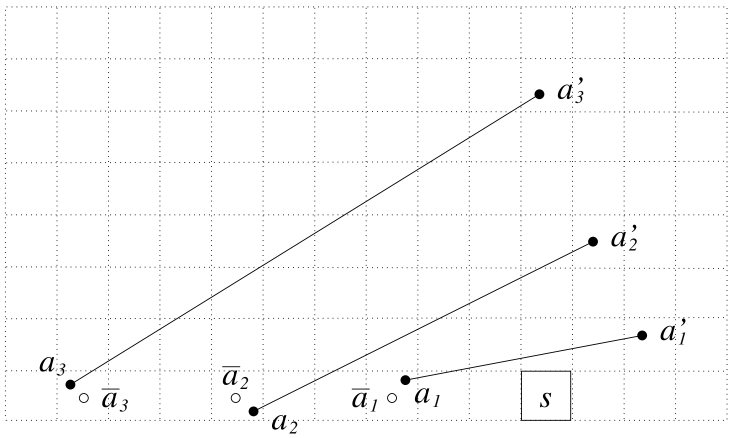

Figure 6.2: Region s is occupied by three messages.

By the definition of k-occupy, $(a_h a'_i) \geq k$ for all h, i and so according to (6.2) we must have for all $h \neq i$ that $(\bar{a}_h a'_i) \geq \sqrt[3]{\alpha}(\bar{a}_h a'_h)$. Thus,

$$(\bar{a}_1 a'_2) \geq \sqrt[3]{\alpha}(\bar{a}_1 a'_1) \tag{6.3}$$

and

$$(\bar{a}_2 a_1') \geq \sqrt[3]{\alpha}(\bar{a}_2 a_2') \tag{6.4}$$

Equation (6.4) and the triangle inequality imply

$$(\bar{a}_2 \bar{a}_1) + (\bar{a}_1 a_1') \geq \sqrt[3]{\alpha}(\bar{a}_2 a_2') \tag{6.5}$$

Multiplying (6.5) by $\sqrt[3]{\alpha}$, and using (6.3), we obtain

$$\alpha^{1/3}(\bar{a}_2 \bar{a}_1) + (\bar{a}_1 a_2') \geq \alpha^{2/3}(\bar{a}_2 a_2') \tag{6.6}$$

However, since the angle $\bar{a}_2 \bar{a}_1 a_2'$ must be obtuse, we see that $(\bar{a}_2 a_2') > (\bar{a}_2 \bar{a}_1)$, and $(\bar{a}_2 a_2') > (\bar{a}_1 a_2')$. Thus, when $\alpha^{2/3} > \alpha^{1/3} + 1$, we arrive at

$$\alpha^{1/3}(\bar{a}_2 \bar{a}_1) + (\bar{a}_1 a_2') < \alpha^{2/3}(\bar{a}_2 a_2') \ .$$

Since this contradicts (6.6), we see that when $\alpha^{2/3} > \alpha^{1/3} + 1$, j can be at most 1.

When α is smaller, we boost the effective value of α in the above equations as follows. Since $(\bar{a}_3 a_2') \geq \sqrt[3]{\alpha}(\bar{a}_3 a_3')$, we have that $(\bar{a}_3 \bar{a}_2) + (\bar{a}_2 a_2') \geq \sqrt[3]{\alpha}(\bar{a}_3 a_3')$. If we multiply this by $\alpha^{2/3}$ and use (6.6) to substitute for $\alpha^{2/3}(\bar{a}_2 a_2')$, we obtain

$$\alpha^{2/3}(\bar{a}_3 \bar{a}_2) + \alpha^{1/3}(\bar{a}_2 \bar{a}_1) + (\bar{a}_1 a_2') \geq \alpha^{2/3}(\bar{a}_3 a_3') \ .$$

Since a_2 and a_3 are at least one region apart, a simple geometric argument gives us that $(\bar{a}_1 a_3') > (\bar{a}_1 a_2')$. Thus we have

$$\alpha^{2/3}(\bar{a}_3 \bar{a}_2) + \alpha^{1/3}(\bar{a}_2 \bar{a}_1) + (\bar{a}_1 a_3') > \alpha(\bar{a}_3 a_3') \ .$$

Iterating the same argument i times, we can show that for any $i \leq j$,

$$\sum_{\ell=1}^{i-1} \alpha^{\ell/3}(\bar{a}_{\ell+1} \bar{a}_\ell) + (\bar{a}_1 a_i') > \alpha^{i/3}(\bar{a}_i a_i') \ .$$

This in turn implies that

$$\alpha^{(i-1)/3}(\bar{a}_i \bar{a}_1) + (\bar{a}_1 a_i') > \alpha^{i/3}(\bar{a}_i a_i'). \tag{6.7}$$

However, we again have that angle $\bar{a}_i \bar{a}_1 a_i'$ is obtuse, and thus that $(\bar{a}_i a_i') > (\bar{a}_i \bar{a}_1)$, and $(\bar{a}_i a_i') > (\bar{a}_1 a_i')$. For any constant $\alpha > 1$, there is a constant I such that $\alpha^{I/3} > 1 + \alpha^{(I-1)/3}$. For $i \geq I$, it must be the case that

$$\alpha^{(i-1)/3}(\bar{a}_i \bar{a}_1) + (\bar{a}_1 a_i') < \alpha^{i/3}(\bar{a}_i a_i').$$

Since this contradicts (6.7), it must be the case that $i < I$, which completes the proof of the lemma. We point out that a more complicated argument using the law of cosines can be used to obtain much better constants. □

This completes the proof of Theorem 6.32. □

We next turn to the case of nodes distributed independently and uniformly at random into any 2-dimensional convex region. As a first step, we consider the problem of nodes distributed randomly in any rectangular region.

Theorem 6.36 *If n nodes are distributed randomly in a rectangular domain space with height H, then the time required to route a random permutation is $\Omega(\frac{n}{H}\min[H, \lceil\frac{\log n}{H}\rceil])$, w.h.p.*

Proof. By using the technique of occupied regions used for the proof of the previous theorem, it is straightforward to show that the time required is $\Omega(\frac{n}{H})$, and thus we only describe the case where $H \le \log n$. The bound for this case follows from the following claim.

Claim 6.37 *There exists a column c between columns $\frac{n}{4H}$ and $\frac{3n}{4H}$ of the n-partition such that w.h.p. at most $O(\lceil\frac{H^2}{\log n}\rceil)$ packets can cross column c at any time step.*

Proof. When $H < \log n$, w.h.p. there is between columns $\frac{n}{4H}$ and $\frac{3n}{4H}$ a set of $\frac{\log n}{4H}$ consecutive columns that do not contain any nodes. To see this, we partition the columns into sets of $\frac{\log n}{4H}$ consecutive columns. Next, we ask the question of how many of these sets between columns $\frac{n}{4H}$ and $\frac{3n}{4H}$ contain at least one node. We then place the nodes, one at a time, uniformly at random into the sets. By using the solution to the coupon collectors problem (see Example 2 in Section 3.2.7 and the comments after Theorem 3.38, or [MR95]), after n nodes have been placed, w.h.p., there is still some set that has not received any nodes.

When $H < \sqrt{\log n}$, at most a constant number of transmissions can be successful across this gap of $\frac{\log n}{4H}$ consecutive columns. When $\sqrt{\log n} < H < \log n$, no more than a constant number of transmissions can be successful across this gap in any set of $\frac{\log n}{H}$ consecutive rows, and thus the total number of transmissions that can be successful is at most $\frac{H^2}{\log n}$. □

Thus, when $H \le \sqrt{\log n}$, the total time required for the transmissions is $\Omega(n)$, and when $\sqrt{\log n} < H < \log n$, the total time required is at least $\Omega(\frac{n \log n}{H^2})$. □

Next we turn our attention to the case of an arbitrarily shaped convex domain space. Note that the convexity assumption gives us that the width W of the domain space is $\Omega(\frac{n}{H})$. We can then use essentially the same lower bound technique as for the case of a rectangular domain space to obtain the following.

Theorem 6.38 *If n nodes are distributed uniformly and independently at random in a convex domain space with height H, then $\Omega(\frac{n}{H}\min[H, \lceil\frac{\log n}{H}\rceil])$ time steps are required to route a random permutation, w.h.p.*

6.4 Proof of Theorem 6.13

6.4.1 The Uniform Case

We first consider the uniform case, that is, all edges have the same latency. Let $\mathcal{S}$ be the offline protocol for $\bar{G}$. We construct from $\mathcal{S}$ a protocol $\mathcal{S}'$ for G that works as follows.

At any time step, a packet waiting to traverse an edge e is given as *rank* the difference between the actual time step and the time step it is supposed to traverse e according to $\mathcal{S}$. If more than one packet want to use the same edge at the same time, then the one with highest rank is preferred for trials to cross it, and the others are *blocked*.

Since we assume $\mathcal{S}$ to be a legal offline protocol (in a way that no two packets are scheduled to traverse the same edge at the same time), we obtain the following fact that will be central for our analysis.

Fact 6.39 *A packet can only be blocked by a packet with strictly higher rank. Furthermore, for any given rank r, any packet of rank more than r can block at most one packet of rank r.*

In the following, we bound the time $\mathcal{S}'$ needs, w.h.p., to simulate $\mathcal{S}$ on G. Let us assume that, for all edges e, $\varphi(e) = \frac{1}{\ell}$ for some fixed $2 \leq \ell \leq L$. Our aim will be to bound the worst case distribution over all ranks of the expected number of packets of certain rank at any time step during the simulation.

Let us first introduce some notation. For any two functions $f_1, f_2 : \mathbb{Z} \to \mathbb{R}_{\geq 0}$, let us say that f_1 *dominates* f_2 (or $f_1 \rhd f_2$) if, for every $i \in \mathbb{Z}$, $\sum_{j \geq i} f_1(j) \geq \sum_{j \geq i} f_2(j)$. We define the *rank distribution* of the N packets at some time step as $r : \mathbb{Z} \to \{0, \ldots, N\}$, where $r(i)$ denotes the current number of packets with rank i, that is, $\sum_i r(i) = N$. (Note that ranks of packets can be negative if packets are ahead of their schedule.) Define the *delay* of a packet at node v to be the current time step minus the dilation of the path it traversed so far. Clearly, during the whole simulation the delay of a packet is an upper bound on its rank. That is, if r denotes the current rank distribution and d denotes the current delay distribution of the packets then, for every $i \in \mathbb{Z}$, $\sum_{j \geq i} d(j) \geq \sum_{j \geq i} r(j)$. Hence, d dominates r. In order to make the proof simpler in the following, assume that any delay change is by ± 1 (in reality, the delays are a multiple of ℓ apart), and any rank change is compressed accordingly.

Let r_0 denote the initial rank distribution and d_0 denote the initial delay distribution of the packets for the simulation of $\mathcal{S}$ by G. Then we define $E_t[r_0]$ to be the expected rank distribution after t time steps of the simulation. Let $\hat{E}_t[d_0]$ denote the expected delay distribution after t time steps, assuming MAXDELAY at each time step, where MAXDELAY is defined as follows:

MAXDELAY: Given a delay distribution d consider, for every i, $\min[\sum_{j>i} d(j), d(i)]$ of the $d(i)$ packets with delay i to be blocked from transmissions.

For every $t \in \mathbb{N}$, $\hat{E}_1^t[d]$ is defined as $\hat{E}_1[\hat{E}_1^{t-1}[d]]$ and $\hat{E}_1^0[d] = d$. Our aim is to show that $\hat{E}_1^t[d_0] \rhd E_t[r_0]$. This would enable us to compute an upper bound for the number of packets with ranks greater than some given rank by applying $\hat{E}_1$ t times to d_0, which is significantly easier to calculate than $E_t[r_0]$. For this, we show the following lemma.

Lemma 6.40 *For every time step $t \in \mathbb{N}_0$, $\hat{E}_1^t[d_0] \rhd E_t[r_0]$.*

Proof. We will show by complete induction that, for all $t \in \mathbb{N}_0$, $\hat{E}_1^t[d_0] \rhd E_t[r_0]$. Clearly, for $t = 0$ the relation is true. Suppose now that the relation holds for some $t \in \mathbb{N}_0$. Let $d = \hat{E}_1^t[d_0]$ and $d' = \hat{E}_1[d]$. Since in case of MAXDELAY an expected amount of $\frac{1}{\ell}(d(i) - \min[d(i), \sum_{j>i} d(j)])$ packets is moved from $d(i)$ to $d(i-1)$ for every $i \in \mathbb{Z}$, we have

$$d'(i) = d(i) - \tfrac{1}{\ell}(d(i) - \min[d(i), \sum_{j>i} d(j)]) + \tfrac{1}{\ell}(d(i+1) - \min[d(i+1), \sum_{j>i+1} d(j)]) \; .$$

Hence, we get for all $i \in \mathbb{Z}$

$$\begin{aligned} \sum_{j \geq i} d'(j) &= \sum_{j \geq i} \left(d(j) - \tfrac{1}{\ell}(d(j) - \min[d(j), \sum_{k>j} d(k)]) + \right. \\ &\qquad \left. \tfrac{1}{\ell}(d(j+1) - \min[d(j+1), \sum_{k>j+1} d(k)]) \right) \\ &= \left(\sum_{j \geq i} d(j) \right) - \tfrac{1}{\ell} d(i) + \tfrac{1}{\ell} \min[d(i), \sum_{j>i} d(j)] \; . \end{aligned} \tag{6.8}$$

Let $A = \{h : [N] \to \mathbb{N}_0\}$ denote the set of all possible assignments of positions to packets in a way that, given an $h \in A$, $h(i)$ denotes the number of edges packet i has already passed. Let the random variable X_t be $h \in A$ if and only if the outcome after t steps of the simulation is h. Given a $h \in A$, let r_h be the corresponding rank distribution. Furthermore, let $r_h^-(i)$ for every $i \in \mathbb{Z}$ denote the number of packets with rank i that are allowed to attempt transmissions, and let $r_h^+(i)$ for every $i \in \mathbb{Z}$ denote the number of packets with rank larger than i that are allowed to attempt transmissions and, in case of success, get rank i. Then it holds for every $i \in \mathbb{Z}$:

$$\begin{aligned} E_{t+1}[r_0](i) &= \sum_{h\in A} \Pr[X_t = h] \cdot (r_h(i) + \tfrac{1}{\ell} r_h^+(i) - \tfrac{1}{\ell} r_h^-(i)) \\ &= E_t[r_0](i) + \tfrac{1}{\ell} \sum_{h\in A} \Pr[X_t = h] \cdot (r_h^+(i) - r_h^-(i)) \;. \end{aligned}$$

Summing up over all $j \geq i$ yields

$$\sum_{j\geq i} E_{t+1}[r_0](j) = \left(\sum_{j\geq i} E_t[r_0](j) \right) + \tfrac{1}{\ell} \sum_{j\geq i} \sum_{h\in A} \Pr[X_t = h] \cdot (r_h^+(j) - r_h^-(j))$$

As it is easy to see, $\sum_{j\geq i} \sum_{h\in A} \Pr[X_t = h] \cdot \frac{1}{\ell}(r_h^+(j) - r_h^-(j))$ represents the expected number of all packets that change from a rank at least i to a rank lower than i after step $t+1$. This number is minimal if we assume that all ranks can only be reduced by one and, for every $h \in A$, $\min[r_h(i), \sum_{j>i} r_h(j)]$ of the packets with rank i are blocked. (The latter argument uses Fact 6.39.) Hence, together with the fact that for any two sequences $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ of real numbers it holds that $\sum_i \min[a_i, b_i] \leq \min[\sum_i a_i, \sum_i b_i]$, we obtain

$$\begin{aligned} \sum_{j\geq i} E_{t+1}[r_0](j) &\leq \left(\sum_{j\geq i} E_t[r_0](j) \right) - \tfrac{1}{\ell} \sum_{h\in A} \Pr[X_t = h] \cdot (r_h(i) - \min[r_h(i),\ \sum_{j>i} r_h(j)]) \\ &= \left(\sum_{j\geq i} E_t[r_0](j) \right) - \tfrac{1}{\ell} E_t[r_0](i) + \\ &\quad \tfrac{1}{\ell} \sum_{h\in A} \Pr[X_t = h] \cdot \min[r_h(i),\ \sum_{j>i} r_h(j)] \\ &\leq \left(\sum_{j\geq i} E_t[r_0](j) \right) - \tfrac{1}{\ell} E_t[r_0](i) + \\ &\quad \tfrac{1}{\ell} \min\left[\sum_{h\in A} \Pr[X_t = h] \cdot r_h(i),\ \sum_{h\in A} \Pr[X_t = h] \cdot \sum_{j>i} r_h(j) \right] \\ &= \left(\sum_{j\geq i} E_t[r_0](j) \right) - \tfrac{1}{\ell} E_t[r_0](i) + \tfrac{1}{\ell} \min[E_t[r_0](i),\ \sum_{j>i} E_t[r_0](j)] \\ &= \sum_{j\geq i} \hat{E}_1[E_t[r_0]](j) \;. \end{aligned} \tag{6.9}$$

It remains to be shown that (6.9) $\leq$ (6.8) for all i. Let $r = E_t[r_0]$. According to the induction hypothesis, $d \triangleright r$. Clearly, from any delay distribution d any rank distribution r that is dominated by d can be constructed by moving units from position i to position $i-1$ for some $i \in \mathbb{Z}$ again and again. (Note that units cannot be added or deleted, since every distribution function f has to fulfill $\sum_i f(i) = N$.) For the case that d is equal to r the claim immediately follows, since ranks of successful packets are reduced by at least as much as their delays. In all other cases we can restrict ourselves to considering two delay distributions d_1 and d_2, where d_2 results from d_1 by moving an ϵ-unit from position i to $i-1$ for some $i \in \mathbb{Z}$. Let $d_1' = \hat{E}_1[d_1]$ and $d_2' = \hat{E}_1[d_2]$. We claim that $\sum_{k\geq j} d_2'(k) \leq \sum_{k\geq j} d_1'(k)$ for all $j \in \mathbb{Z}$. For this, we have to consider some cases.

- For all $j < i-1$ and $j > i$ it clearly holds that $d_1(j) = d_2(j)$ and $\sum_{k>j} d_1(k) = \sum_{k>j} d_2(k)$. Hence, $\sum_{k\geq j} d_1'(k) = \sum_{k\geq j} d_2'(k)$, which fulfills the claim for these j.
- For $j = i$ it holds that $d_1(j) = d_2(j) + \epsilon$ and $\sum_{k>j} d_1(k) = \sum_{k>j} d_2(k)$. Hence, $\sum_{k\geq j} d_1'(k) \geq \sum_{k\geq j} d_2'(k)$, which also fulfills the claim for this j.
- For $j = i-1$, we have $d_2(j) = d_1(j) + \epsilon$ and $\sum_{k>j} d_2(k) = \sum_{k>j} d_1(k) - \epsilon$. If $d_1(j) \leq \sum_{k>j} d_1(k)$ then, clearly, $\sum_{k\geq j} d_1'(k) \geq \sum_{k\geq j} d_2'(k)$. It therefore remains to consider the case $d_1(j) > \sum_{k>j} d_1(k)$. Let $\delta = d_1(j) - \sum_{k>j} d_1(k)$. Then $\sum_{k\geq j} d_1'(k) = \sum_{k\geq j} d_1(k) - \delta/\ell$. On the other hand,

  $$d_2(j) - \sum_{k>j} d_2(k) = (d_1(j) + \epsilon) - (\sum_{k>j} d_1(k) - \epsilon) = \delta + 2\epsilon \ .$$

  Hence,

  $$\sum_{k\geq j} d_2'(k) = \sum_{k\geq j} d_2(k) - (\delta + 2\epsilon)/\ell = \sum_{k\geq j} d_1(k) - (\delta + 2\epsilon)/\ell \leq \sum_{k\geq j} d_1'(k) \ .$$

  Thus, the claim is also true for $j = i-1$.

This completes the induction step, which proves the lemma. □

Lemma 6.40 implies that we can use the same assumption for the delays of the packets as for their ranks, formulated in Fact 6.39: packets can only be blocked by packets with higher delays, and for every given delay d, any packet with a delay of more than d can block at most one packet with delay d. Furthermore, the lemma implies that in order to have an upper bound for $\sum_{j\geq i} E_t[r_0](j)$ at some time step t it remains to find an upper bound for $\sum_{j\geq i} \hat{E}_1^t[d_0](j)$.

Since we assume all edges e to have a success probability of $1/\ell$ for some fixed ℓ, every time a packet is successful, its delay is decreased by $\ell - 1$, and every time it is not successful, its delay is increased by one. Hence, at any time step, the difference between the delays of any two packets is an integer multiple of ℓ. In order to simplify the analysis for the distribution of the packets, let us consider instead of the delay the number of edges a packet already passed. Clearly, if at time step t a packet already passed s edges then its delay is $t - \ell \cdot s$.

For every time step t, let the distribution function $p_t : \mathbb{N}_0 \to [0,1]$ be defined such that, for all s, $p_t(s) \cdot N$ denotes the expected number of packets that have already passed s edges. Initially, $p_0(0) = 1$ and $p_0(s) = 0$ for all $s > 0$. In the next lemma, we show that p_t can be bounded as follows.

Lemma 6.41 *Using MAXDELAY at every time step $t \geq 1$ it holds that, for all $s \geq 0$,*

$$p_t(s) \leq \frac{\left(\frac{2t}{\ell-1}\right)^s \left(1 - \frac{1}{\ell}\right)^t}{s!} \ .$$

Proof. We show this by complete induction over t. It is easy to check that for $t = 1$ the bound is correct. So suppose that for some $t \geq 1$ it has already been shown that the bound above is correct. Then we show that it is also correct for $t + 1$. For this we use the fact that for an upper bound we are allowed to apply $\hat{E}_1$ for any time step t. Let us consider the following cases in order to bound $p_{t+1}(s)$ for all $s \in \mathbb{N}_0$.

Case $p_t(s) \geq \sum_{s'<s} p_t(s')$ **and** $p_t(s-1) \geq \sum_{s'<s-1} p_t(s')$**:**
In this case, using MAXDELAY, an expected fraction of $\frac{1}{\ell}(p_t(s) - \sum_{s'<s} p_t(s'))$ is moved from $p_t(s)$ to $p_t(s+1)$, while an expected fraction of $\frac{1}{\ell}(p_t(s-1) - \sum_{s'<s-1} p_t(s'))$ is moved from $p_t(s-1)$ to $p_t(s)$. Hence, we get

$$\begin{aligned} p_{t+1}(s) &= p_t(s) - \tfrac{1}{\ell}(p_t(s) - \sum_{s'<s} p_t(s')) + \tfrac{1}{\ell}(p_t(s-1) - \sum_{s'<s-1} p_t(s')) \\ &= \left(1 - \tfrac{1}{\ell}\right)p_t(s) + \tfrac{2}{\ell}p_t(s-1) \ . \end{aligned}$$

Using the induction hypothesis, it follows that

$$\begin{aligned} p_{t+1}(s) &\leq \left(1 - \frac{1}{\ell}\right) \cdot \frac{\left(\frac{2t}{\ell-1}\right)^s \left(1 - \frac{1}{\ell}\right)^t}{s!} + \frac{2}{\ell} \cdot \frac{\left(\frac{2t}{\ell-1}\right)^{s-1} \left(1 - \frac{1}{\ell}\right)^t}{(s-1)!} \\ &\leq \frac{\left(\frac{2t}{\ell-1}\right)^s \left(1 - \frac{1}{\ell}\right)^{t+1}}{s!} \left(1 + \frac{s}{t}\right) \ . \end{aligned}$$

It holds that

$$\begin{aligned} \frac{\left(\frac{2t}{\ell-1}\right)^s \left(1 - \frac{1}{\ell}\right)^{t+1}}{s!} \left(1 + \frac{s}{t}\right) &\leq \frac{\left(\frac{2(t+1)}{\ell-1}\right)^s \left(1 - \frac{1}{\ell}\right)^{t+1}}{s!} \\ \Leftrightarrow \qquad 1 + \frac{s}{t} &\leq \left(1 + \frac{1}{t}\right)^s \ , \end{aligned}$$

which is clearly true for any $s \geq 0$ and $t \geq 1$. This completes the first case.

Case $p_t(s) < \sum_{s'<s} p_t(s')$ **and** $p_t(s-1) \geq \sum_{s'<s-1} p_t(s')$**:**
In this case, using MAXDELAY, an expected fraction of $\frac{1}{\ell}(p_t(s-1) - \sum_{s'<s-1} p_t(s'))$ is moved from $p_t(s-1)$ to $p_t(s)$. Hence, we get

$$p_{t+1}(s) = p_t(s) + \tfrac{1}{\ell}(p_t(s-1) - \sum_{s'<s-1} p_t(s')) \ .$$

This term is less than $(1 - \frac{1}{\ell})p_t(s) + \frac{2}{\ell}p_t(s-1)$ if and only if

$$\begin{aligned} \tfrac{1}{\ell}p_t(s) &< \tfrac{1}{\ell}p_t(s-1) + \tfrac{1}{\ell} \sum_{s'<s-1} p_t(s') \\ \Leftrightarrow \quad p_t(s) &< p_t(s-1) + \sum_{s'<s-1} p_t(s') \ , \end{aligned}$$

which is true since $p_t(s) < \sum_{s'<s} p_t(s')$. Thus, we can use the same calculation as above to prove the bound for $p_{t+1}(s)$ also in this case.

Case $p_t(s-1) < \sum_{s'<s-1} p_t(s')$**:**
Then it is easy to see that nothing can have reached $p_t(s)$ yet, that is, $p_t(s) = 0$. This completes the proof of Lemma 6.41. □

Since each successful traversal of an edge by a packet corresponds to ℓ steps in $\mathcal{S}$, $d_t(s) = p_t(s/\ell) \cdot N$ corresponds to the expected number of packets that are at time step t of the simulation at step s in $\mathcal{S}$, assuming MAXDELAY. Clearly, the simulation finishes if all packets are beyond step T in $\mathcal{S}$. It therefore remains to be shown that for $t = O(T + L\log N)$ this is the case w.h.p.

Let the random variable X be defined as the number of packets at time step t of the simulation with a rank of at least $t - T$, i.e., that are at step T or before in $\mathcal{S}$. Due to Lemma 6.40, $\mathrm{E}[X]$ can be upper bounded by $\sum_{\delta \geq t-T} d_t(t-\delta)$. Since during the simulation the delays of the packets are always a multiple of ℓ apart, it follows for any t that is a multiple of ℓ that

$$\mathrm{E}[X] \leq \sum_{\substack{t-T\leq\delta\leq t,\\ \ell|(t-\delta)}} d_t(t-\delta) = N \sum_{\substack{t-T\leq\delta\leq t,\\ \ell|(t-\delta)}} p_t\left(\frac{t-\delta}{\ell}\right) . \tag{6.10}$$

In order to bound this sum, we first bound $p_t(s/\ell)$ for $s = T$ and $t = (1+\epsilon)T$. If $\epsilon \geq 4$ then

$$\begin{aligned} p_t(s/\ell) = \frac{\left(\frac{2(1+\epsilon)T}{\ell-1}\right)^{T/\ell}\left(1-\frac{1}{\ell}\right)^{(1+\epsilon)T}}{(T/\ell)!} &\leq \frac{\left(\frac{T}{\ell-1}\right)^{T/\ell}}{\mathrm{e}^{T/\ell}(T/\ell)!}\left(\frac{2(1+\epsilon)}{\mathrm{e}^{\epsilon}}\right)^{T/\ell} \\ &\leq \left(\frac{\ell}{\ell-1}\right)^{T/\ell}\left(\frac{2(1+\epsilon)}{\mathrm{e}^{\epsilon}}\right)^{T/\ell} \\ &\leq \mathrm{e}^{\frac{T}{\ell}\left(\frac{1}{\ell-1}-\frac{\epsilon}{3}\right)} . \end{aligned} \tag{6.11}$$

Furthermore, it is easy to check that

$$p_t(s') \leq p_t(s'+1) \quad \Leftrightarrow \quad s'+1 \leq \frac{2t}{\ell-1} , \tag{6.12}$$

which is true for all $s' \leq t/\ell$ and $\ell \geq 2$. Using (6.11) and (6.12) in (6.10), we obtain

$$\mathrm{E}[X] \leq N \cdot \frac{\epsilon T}{\ell} \cdot \mathrm{e}^{\frac{T}{\ell}\left(\frac{1}{\ell-1}-\frac{\epsilon}{3}\right)} .$$

This can be made less than $N^{-\alpha}$ for any constant $\alpha > 0$ if $\epsilon \geq 4$ is sufficiently large. Assuming that $\mathrm{E}[X] \leq N^{-\alpha}$, it follows from the Markov inequality that

$$\Pr[X \geq 1] \leq \left(\frac{1}{N}\right)^{\alpha} .$$

This concludes the proof of Theorem 6.13 for the uniform case.

6.4.2 The Non-Uniform Case

Now, we show how to prove Theorem 6.13 also for the non-uniform case. Although we only show it for PCGs, it also holds for any transmission graph using a zone access scheme if we assume that at every time step all zones at all nodes participate in decisions to attempt hops, whether a packet has to be sent out for some zone or not. This ensures that the probability distribution for certain node–zone pairs to have or not to have success is the same throughout the whole simulation. We will note at specific places in our proof where we need this assumption.

Let $\mathcal{S}_1$ be an offline protocol for some DCG $\bar{G}$. First, we modify $\mathcal{S}_1$ so that it is easier to bound the time of simulating it. We start with transforming $\mathcal{S}_1$ into a schedule $\mathcal{S}_2$ in which all edge latencies are powers of 2 and every edge with latency ℓ only starts to send a packet at an integer multiple of ℓ. This can be done as follows:

Expand $\mathcal{S}_1$ by a factor of two in a way that an edge that previously forwarded a packet at time t now starts to forward it at time $2t$. Change the latency of every edge to the next highest power of two. Thus, the highest and the lowest edge latency, L and ℓ, are now powers of two. Clearly, the new schedule is still legal, in a sense that traversals of edges by a packet do not overlap in time, and the order of transmitting packets at the edges is preserved. Expand $\mathcal{S}_1$ again by a factor of two as before. Change the starting time of every traversal to the next highest time that is an integer multiple of the corresponding edge latency. Again it is easy to see that the resulting schedule is legal and the order of transmissions is preserved.

Next, transform $\mathcal{S}_2$ into a schedule $\mathcal{S}_3$ where all packets use paths of the same type p. This path p consists of a so-called *(L,ℓ)-sequence s* of $2L/\ell - 1$ edges that repeats itself again and again. The edge latency of edge $i \in \{1,\ldots,2L/\ell-1\}$ of the (L,ℓ)-sequence is $\ell \cdot j$, where j is the largest integer for which i is an integer multiple of 2^{j-1}. To give an example, for $\ell = 1$ and $L = 8$, s represents a sequence of edges with latencies

$$1,2,1,4,1,2,1,8,1,2,1,4,1,2,1.$$

It can be easily checked that the time to traverse all edges of the (L,ℓ)-sequence one after the other is equal to $L\log(2L/\ell)$. For each such sequence, $\mathcal{S}_3$ reserves a time interval of length $L\log(2L/\ell)$ so that disjoint time intervals can be assigned to the edges.

In order to transform $\mathcal{S}_2$ into $\mathcal{S}_3$, consider any time interval I of length L in $\mathcal{S}_2$ starting at time step t, where t is an integer multiple of L. Let us map $\mathcal{S}_2$ restricted to I to $\mathcal{S}_3$ restricted to the $(t/L+1)$st occurrence of s in p, denoted by s', in the following way: for every packet P and every edge e with latency ℓ' scheduled to be used by P with starting time t in I, schedule P in $\mathcal{S}_3$ to use e represented by the $(\frac{t}{\ell'}+1)$st edge in s' with latency ℓ'. (This is possible, since t is at most $L-\ell'$ and an integer multiple of ℓ'.) As it is easy to check, the transformation ensures that

- traversals of different edges by the same packet and
- the traversal of the same edge by different packets

remain in order and do not overlap. In addition to this transformation, we fill all time intervals a packet is not scheduled to traverse an edge by traversals of the corresponding edges in p. For any packet P, these edges are called *imaginary edges*, whereas the edges already used in $\mathcal{S}_2$ are called *real edges*. So we end up with a schedule $\mathcal{S}_3$ in which all packets traverse the same type of path p for T/L occurrences of s without waiting, where T is the runtime of $\mathcal{S}_2$.

$\mathcal{S}_3$ will be simulated by the PCG G of $\bar{G}$ in the following way: The traversal of any real edge e by some packet, where e corresponds to the original edge e' in $\bar{G}$, is simulated in $\bar{G}$ by attempts to cross the counterpart of e', e'', in G. If e has a larger latency than e', then we add an extra random experiment that decides whether to attempt to traverse e'' so that, together with the probability that the attempt fails for e'', we arrive at a probability of traversing e'' that is the inverse of the latency of e. (Recall that for transmission graphs using zone access schemes, all probabilities are already powers of 2. Hence, an offline protocol for its corresponding DCG does not require any change in the latency of the edges.) The traversal of any imaginary edge by some packet is simply simulated *internally* in the PCG, that is, the corresponding packet does not leave its current node. Obviously, once the PCG finishes $\mathcal{S}_3$ it has also finished $\mathcal{S}_1$. Hence, it remains to provide an upper bound for the time the PCG requires to simulate $\mathcal{S}_3$. For this we show the following lemma.

Lemma 6.42 *For every time step $t \in \mathbb{N}_0$, $\hat{E}_t[d_0] \triangleright E_t[r_0]$.*

Proof. We present here a different proof than for Lemma 6.40. The reason for this is that, in the uniform case, ranks are decreased always by at least as much as delays, no matter where some packet might be for two different simulations. This does not hold for the nonuniform case.

Let the rank and the delay of a packet during the course of the simulation be defined as for the uniform case. Since in $\mathcal{S}_3$ no packet is ever waiting at an edge, the following fact holds.

Fact 6.43 *Schedule $\mathcal{S}_3$ has the property that, at any time during its simulation, the delay of a packet is always equal to its rank.*

To continue, we need some notation. For any two multisets D_1, D_2 of functions of the form $f : [N] \to \mathbb{Z}$ with the property that $|D_1| = |D_2|$ we say that D_1 *strongly dominates* D_2 (or $D_1 \rhd_s D_2$) if there is a permutation $\pi : D_1 \to D_2$ such that for all $d \in D_1$ it holds that $d(i) \geq \pi(d)(i)$ (or $d \rhd_s \pi(d)$).

Let $P_t[d_0]$ be defined as the probability distribution over all assignments of delays to the packets after t steps of the simulation. Similarly, let $\hat{P}_t[d_0]$ be defined as the probability distribution over all assignments of delays to the packets after t steps, assuming MAXDELAY at every step. However, instead of allowing to choose an arbitrary set of packets as in the uniform case, we will select a specific set of packets, depending on the given situation. Since the success probability of any packet at any node is a rational number, we can transform $P_t[d_0]$ into a multiset $D_t[d_0]$ with the property that, for all assignments $d : [N] \to \mathbb{Z}$ of delays to the N packets, $P_t[d_0](d) =$ (number of times d appears in $D_t[d_0])/|D_t[d_0]|$. $\hat{D}_t[d_0]$ is defined accordingly, using $\hat{P}_t[d_0]$. Let $D_t[d_0]$ and $\hat{D}_t[d_0]$ be chosen such that they are of the same cardinality.

Our aim is to show by complete induction that, for all $t \in \mathbb{N}_0$, $\hat{D}_t[d_0] \rhd_s D_t[d_0]$. Clearly, if $\hat{D}_t[d_0] \rhd_s D_t[d_0]$ is true, then this would immediately imply that $\hat{E}_t[d_0] \rhd E_t[d_0]$. Since, by Fact 6.43, during the whole simulation the delay of a packet is equal to its rank, $E_t[d_0] = E_t[r_0]$ and therefore the lemma would follow.

For $t = 0$ our claim is obviously true. Suppose now that $\hat{D}_t[d_0]$ strongly dominates $D_t[d_0]$ for some $t \in \mathbb{N}_0$. Let the permutation $\pi : \hat{D}_t[d_0] \to D_t[d_0]$ be defined such that, for all $d \in \hat{D}_t[d_0]$, $d \rhd_s \pi(d)$. Fix any $d \in \hat{D}_t[d_0]$, and let $d' = \pi(d)$. We will show that then $\hat{D}_1[d] \rhd_s D_1[d']$. From this the lemma would follow.

According to the definition of $\rhd_s$, for every packet P it holds that P w.r.t. d is not ahead of P w.r.t. d'. (Notice that once the delay of a packet is known at a specific time step, then also its position is known.) Clearly, all packets whose positions w.r.t. d are behind their positions w.r.t. d' imply $\hat{D}_1[d] \rhd_s D_1[d']$ restricted to these packets. So we only have to consider those packets whose positions are the same w.r.t. d and d'. As Fact 6.39 holds for $\mathcal{S}_3$ and the delays are equal to the ranks, MAXDELAY is the worst case that can happen for the packets placed according to d'. Since, due to $d \rhd_s d'$, $|\{j \mid d(j) \geq i\}| \geq |\{j \mid d'(j) \geq i\}|$ for all $i \in \mathbb{Z}$, all packets with equal positions w.r.t. d and d' that are blocked w.r.t. d' can also be blocked via MAXDELAY w.r.t. d. Hence, $\hat{D}_1[d] \rhd_s D_1[d']$ restricted to all packets that are blocked w.r.t. d'. For the remaining packets we have the situation that w.r.t. d and d' they are allowed to perform access trials, that is, the probability distribution for any combination of these packets to be successfully transmitted is the same for both situations. (Note that we need here the assumption formulated at the beginning of the section that in a transmission graph all zones in all nodes participate in decisions for hop trials. Otherwise our results for PCGs cannot be transferred to the transmission graphs, since the probability distributions are not the same.) This implies that $\hat{D}_1[d] \rhd_s D_1[d']$ for all packets, which proves the lemma. □

Furthermore, the following lemma holds.

Lemma 6.44 *For every time step $t \in \mathbb{N}_0$, $\hat{E}_1^t[d_0] \triangleright \hat{E}_t[d_0]$.*

Proof. We will prove the lemma by induction on t. For $t = 0$ the lemma is obviously true. So suppose in the following that it has already been shown for some $t \geq 0$. Let $d = \hat{E}_1^t[d_0]$ and $d' = \hat{E}_1[d]$. To simplify the arguments, assume that instead of regarding the delay as the sum of the latencies of the edges the packet is behind (or ahead of) its schedule, we see the delay simply as the sum of the edges the packet is behind (or ahead of) its schedule. We can use this simplification, since all packets use exactly the same type of path and all packets intend to traverse this type of path without waiting, which means that a fixed delay specifies a fixed edge of the path at which the packet can be currently found.

Assume that, for the given time step t, a delay of i specifies edge e_i. Let ℓ_i denote the latency of e_i. In this case, it follows analogous to the proof of Lemma 6.40 that, for every $i \in \mathbb{Z}$, we have

$$\begin{aligned}
\sum_{j \geq i} d'(j) &= \sum_{j \geq i} \left(d(j) - \tfrac{1}{\ell_j}(d(j) - \min[d(j), \sum_{k>j} d(k)]) + \right. \\
&\qquad \left. \tfrac{1}{\ell_{j+1}}(d(j+1) - \min[d(j+1), \sum_{k>j+1} d(k)]) \right) \\
&= \left(\sum_{j \geq i} d(j) \right) - \tfrac{1}{\ell_i} d(i) + \tfrac{1}{\ell_i} \min[d(i), \sum_{j>i} d(j)] \ .
\end{aligned}$$

Next we calculate $\hat{E}_{t+1}[d_0]$. Let $D = \{h : [N] \to \mathbb{Z}\}$ denote the set of all possible assignments of delays to packets, i.e., $h(i)$ denotes the delay of packet i. Moreover, let the random variable X_t be $h \in D$ if and only if the outcome after t steps is h. Given a $h \in D$, let d_h denote the corresponding delay distribution, i.e., $d_h(i)$ is equal to the number of packets with delay i. Since the MAXDELAY rule is applied to the packets, we have

$$\begin{aligned}
\sum_{j \geq i} \hat{E}_{t+1}[d_0](j) &= \sum_{h \in D} \Pr[X_t = h] \sum_{j \geq i} \left(d(j) - \tfrac{1}{\ell_j}(d(j) - \min[d(j), \sum_{k>j} d(k)]) + \right. \\
&\qquad \left. \tfrac{1}{\ell_{j+1}}(d(j+1) - \min[d(j+1), \sum_{k>j+1} d(k)]) \right) \\
&= \left(\sum_{j \geq i} \hat{E}_t[d_0](j) \right) - \tfrac{1}{\ell_i} \hat{E}_t[d_0](i) + \tfrac{1}{\ell_i} \sum_{h \in D} \Pr[X_t = h] \cdot \min[d(i), \sum_{j>i} d(j)] \\
&\leq \left(\sum_{j \geq i} \hat{E}_t[d_0](j) \right) - \tfrac{1}{\ell_i} \hat{E}_t[d_0](i) + \\
&\qquad \tfrac{1}{\ell_i} \min \left[\sum_{h \in D} \Pr[X_t = h] \cdot d(i), \sum_{h \in D} \Pr[X_t = h] \sum_{j>i} d(j) \right] \\
&= \left(\sum_{j \geq i} \hat{E}_t[d_0](j) \right) - \tfrac{1}{\ell_i} \hat{E}_t[d_0](i) + \tfrac{1}{\ell_i} \min[\hat{E}_t[d_0](i), \sum_{j>i} \hat{E}_t[d_0](j)] \\
&= \sum_{j \geq i} \hat{E}_1[\hat{E}_t[d_0](j)] \ .
\end{aligned}$$

Since, by the induction hypothesis, $\hat{E}_1^t[d_0] \triangleright \hat{E}_t[d_0]$, it follows similar to the proof of Lemma 6.40 that also $\hat{E}_1^{t+1}[d_0] \triangleright \hat{E}_{t+1}[d_0]$. $\square$

Lemmas 6.42 and 6.44 allow us to argue in the following only with $\hat{E}_1^t[d_0]$. For a final transformation from $\mathcal{S}_3$ to a schedule $\mathcal{S}_4$, we need the following lemma.

Lemma 6.45 *Consider any schedule $\mathcal{S}$ in which all packets follow a path of the same type p. Let $\mathcal{S}'$ be a schedule in which the same number of packets follow a path of the same type p'. p' results from p by exchanging the positions of two consecutive edges in p, where the first edge has a lower latency than the second edge. Then $\hat{E}_1^t[d_0]$ w.r.t. $\mathcal{S}'$ dominates $\hat{E}_1^t[d_0]$ w.r.t. $\mathcal{S}$.*

Proof. Let us denote the two edges to be exchanged as e_1 and e_2, where e_1 be the ith edge of p. (W.l.o.g. we assume that $i \geq 2$. The case $i = 1$ can be prevented by inserting an edge with latency L in front of p.) Let ℓ_1 be the latency of e_1 and ℓ_2 be the latency of e_2, $\ell_1 < \ell_2$. Given a time step t, let $\boldsymbol{a}^{(t)}, \boldsymbol{x}^{(t)} \in \mathbb{R}^4$ be two vectors, where

- $a_0^{(t)}$ (resp. $x_0^{(t)}$) denotes the expected number of packets that have not traversed the $(i-1)$st edge in p (resp. p') yet,
- $a_1^{(t)}$ (resp. $x_1^{(t)}$) represents the expected number of packets that is currently at the ith edge of p (resp. p'),
- $a_2^{(t)}$ (resp. $x_2^{(t)}$) is the expected number of packets that is currently at the $(i+1)$st edge of p (resp. p'), and
- $a_3^{(t)}$ (resp. $x_3^{(t)}$) denotes the expected number of packets that has already passed the $(i+1)$st edge of p (resp. p').

See also Figure 6.3.

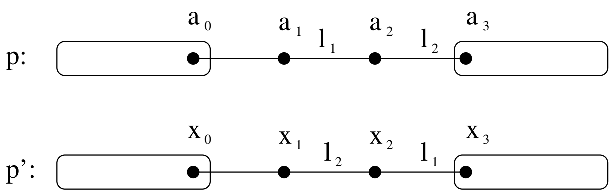

Figure 6.3: The positions of the a_i and x_i in the two paths p and p'

We define $\boldsymbol{x}^{(t)}$ to *dominate* $\boldsymbol{a}^{(t)}$ (or $\boldsymbol{x}^{(t)} \rhd \boldsymbol{a}^{(t)}$) if for all i we have $\sum_{j \leq i} x_i^{(t)} \geq \sum_{j \leq i} a_i^{(t)}$. If we could show that under MAXDELAY we have $\boldsymbol{x}^{(t)} \rhd \boldsymbol{a}^{(t)}$ for all $t \geq 0$ then the lemma would obviously follow.

Clearly, it holds $a_0^{(0)} = x_0^{(0)} = 1$ and $a_j^{(0)} = x_j^{(0)} = 0$ for all $j \in \{1, 2, 3\}$. So for $t = 0$, $\boldsymbol{x}^{(t)} \rhd \boldsymbol{a}^{(t)}$. For the remaining steps it remains to prove the following proposition.

Proposition 6.46 *For any strategy of moving packets from $a_0^{(t)}$ to $a_1^{(t)}$ that is also used for moving packets from $x_0^{(t)}$ to $x_1^{(t)}$ at every time step $t \in \mathbb{N}$ it holds under MAXDELAY that $\boldsymbol{x}^{(t)} \rhd \boldsymbol{a}^{(t)}$ for all $t \in \mathbb{N}$.*

Proof. For any $t \in \mathbb{N}$, let ϵ_t denote the amount of load moved from $a_0^{(t)}$ to $a_1^{(t)}$ (resp. from $x_0^{(t)}$ to $x_1^{(t)}$). As long as $a_1^{(t)} \leq a_0^{(t)}$ it clearly holds $a_i^{(t)} = x_i^{(t)}$ for all $i \in \{1, 2, 3\}$, so $\boldsymbol{x}^{(t)} \rhd \boldsymbol{a}^{(t)}$. Let t_1 be the first time step for which $a_1^{(t_1)} > a_0^{(t_1)}$, and let t_2 be the first time step for which

$a_2^{(t_2)} > a_0^{(t_2)} + a_1^{(t_2)}$. Since $\ell_1 < \ell_2$, it holds $a_2^{(t)} > x_2^{(t)}$ and $a_3^{(t)} = x_3^{(t)} = 0$ for all $t_1 < t \leq t_2$. Hence, $\boldsymbol{x}^{(t)} \triangleright \boldsymbol{a}^{(t)}$ also for $t_1 \leq t \leq t_2$.

For $t = t_2 + 1$, we have $a_3^{(t)} > 0$ for the first time. Let ϵ be chosen such that $a_3^{(t_2+1)} = \frac{4}{\ell_1 \ell_2}\epsilon$. This means that $a_2^{(t_2)} = a_0^{(t_2)} + a_1^{(t_2)} + \frac{4}{\ell_1}\epsilon$, and because of $a_2^{(t_2-1)} < a_0^{(t_2-1)} + a_1^{(t_2-1)}$ that $a_1^{(t_2-1)} = a_0^{(t_2-1)} + 4\epsilon + \ell_1\delta$ and $a_2^{(t_2-1)} = a_0^{(t_2-1)} + a_1^{(t_2-1)} - \delta$ for some $\delta > 0$. Let the vectors $\boldsymbol{b}^{(t)}, \boldsymbol{c}^{(t)} \in \mathbb{R}^4$ be defined as $\boldsymbol{b}^{(t_2-1)} = \boldsymbol{a}^{(t_2-1)} + (\epsilon, -\epsilon, 0, 0)$ and $\boldsymbol{c}^{(t_2-1)} = (-\epsilon, \epsilon, 0, 0)$, that is, $\boldsymbol{a}^{(t_2-1)} = \boldsymbol{b}^{(t_2-1)} + \boldsymbol{c}^{(t_2-1)}$. For all $t \geq t_2$, let the load moved from $a_0^{(t)}$ to $a_1^{(t)}$ be equal to the load moved from $b_0^{(t)}$ to $b_1^{(t)}$. Let MAXDELAY be applied to $\boldsymbol{b}^{(t)}$ and $\boldsymbol{c}^{(t)}$. Then the equations for $\boldsymbol{a}^{(t_2-1)}$ above yield that

- $\boldsymbol{b}^{(t_2)} = (a_0^{(t_2)} + \epsilon, a_1^{(t_2)} - (1 - \frac{2}{\ell_1})\epsilon, a_2^{(t_2)} - \frac{2}{\ell_1}\epsilon, 0)$ and
- $\boldsymbol{c}^{(t_2)} = (-\epsilon, (1 - \frac{2}{\ell_1})\epsilon, \frac{2}{\ell_1}\epsilon, 0)$.

Hence, $\boldsymbol{a}^{(t_2)} = \boldsymbol{b}^{(t_2)} + \boldsymbol{c}^{(t_2)}$. Furthermore, it is easy to check that $b_2^{(t_2)} = b_0^{(t_2)} + b_1^{(t_2)}$, using the equation for $a_2^{(t_2)}$ above. Thus Claim 6.47 shows that $\boldsymbol{a}^{(t)} = \boldsymbol{b}^{(t)} + \boldsymbol{c}^{(t)}$ also for all $t > t_2$.

Claim 6.47 *Given some time step t with vectors $\boldsymbol{a}^{(t)}$, $\boldsymbol{b}^{(t)}$ and $\boldsymbol{c}^{(t)}$, where*

- $a_i^{(t)} = b_i^{(t)} + c_i^{(t)}$ *for all $i \in \{0, \ldots, 3\}$,*
- $a_1^{(t)} \geq a_0^{(t)}$, $a_2^{(t)} \geq a_1^{(t)} + a_0^{(t)}$,
- $b_1^{(t)} \geq b_0^{(t)}$, $b_2^{(t)} \geq b_1^{(t)} + b_0^{(t)}$,
- $c_1^{(t)} \geq c_0^{(t)}$, $c_2^{(t)} \geq c_1^{(t)} + c_0^{(t)}$, *and*
- *a load of $\epsilon \geq 0$ is moved from $a_0^{(t)}$ to $a_1^{(t)}$ and from $b_0^{(t)}$ to $b_1^{(t)}$ in step $t + 1$.*

Then it holds that, using MAXDELAY for $a^{(t)}$, $b^{(t)}$ and $c^{(t)}$,

- $a_i^{(t+1)} = b_i^{(t+1)} + c_i^{(t+1)}$ *for all $i \in \{0, \ldots, 3\}$,*
- $a_1^{(t+1)} \geq a_0^{(t+1)}$, $a_2^{(t+1)} \geq a_1^{(t+1)} + a_0^{(t+1)}$,
- $b_1^{(t+1)} \geq b_0^{(t+1)}$, $b_2^{(t+1)} \geq b_1^{(t+1)} + b_0^{(t+1)}$, *and*
- $c_1^{(t+1)} \geq c_0^{(t+1)}$, $c_2^{(t+1)} \geq c_1^{(t+1)} + c_0^{(t+1)}$

Proof. Follows directly from the application of MAXDELAY. □

Similarly, let us decompose $\boldsymbol{x}^{(t)}$ into $\boldsymbol{y}^{(t)}$ and $\boldsymbol{z}^{(t)}$ with $\boldsymbol{y}^{(t_2-1)} = \boldsymbol{x}^{(t_2-1)} + (\epsilon, -\epsilon, 0, 0)$ and $\boldsymbol{z}^{(t_2-1)} = (-\epsilon, \epsilon, 0, 0)$. Then the following claim holds.

Claim 6.48 *Under the assumption that MAXDELAY is applied to $\boldsymbol{x}^{(t)}$, $\boldsymbol{y}^{(t)}$ and $\boldsymbol{z}^{(t)}$ and that the load moved from $x_0^{(t)}$ to $x_1^{(t)}$ is equal to the load moved from $y_0^{(t)}$ to $y_1^{(t)}$ for all t, it holds that $\boldsymbol{x}^{(t)} \triangleright \boldsymbol{y}^{(t)} + \boldsymbol{z}^{(t)}$ for all $t \geq t_2 - 1$.*

Proof. Let t_3 be the first time step for which $y_2^{(t)} \geq y_1^{(t)} + y_0^{(t)}$. Clearly, in this case it already holds that $x_2^{(t)} \geq x_1^{(t)} + x_0^{(t)}$. Suppose that $\boldsymbol{x}^{(t_3)} \triangleright \boldsymbol{y}^{(t_3)} + \boldsymbol{z}^{(t_3)}$. Define $\boldsymbol{u}^{(t_3)} = \boldsymbol{y}^{(t_3)} + \boldsymbol{z}^{(t_3)}$. Then, according to Claim 6.47, $\boldsymbol{u}^{(t)} = \boldsymbol{y}^{(t)} + \boldsymbol{z}^{(t)}$ for all $t \geq t_3$. Since $\boldsymbol{x}^{(t_3)} \triangleright \boldsymbol{u}^{(t_3)}$, applying MAXDELAY ensures that $\boldsymbol{x}^{(t)} \triangleright \boldsymbol{u}^{(t)}$ for all $t \geq t_3$. Hence, the claim would be true for all $t \geq t_3$, given that $\boldsymbol{x}^{(t_3)} \triangleright \boldsymbol{y}^{(t_3)} + \boldsymbol{z}^{(t_3)}$.

We prove the claim for $t_2 - 1 \leq t \leq t_3$ by complete induction over t. For $t = t_2 - 1$ the claim is obviously true. Suppose now that the claim is true for some $t_2 - 1 \leq t < t_3$. Let $\boldsymbol{u}^{(t)} = (z_0^{(t)}, z_1^{(t)}, x_2^{(t)} - y_2^{(t)}, z_3^{(t)} + (z_2^{(t)} - (x_2^{(t)} - y_2^{(t)})))$. Since it follows from Claim 6.47 that $x_0^{(t')} = y_0^{(t')} + z_0^{(t')}$ and $x_1^{(t')} = y_1^{(t')} + z_1^{(t')}$ for all $t' \geq t_2 - 1$, it holds that $x_2^{(t)} + x_3^{(t)} = y_2^{(t)} + z_2^{(t)} + z_3^{(t)}$. Hence,

$$x_2^{(t)} - y_2^{(t)} \geq z_2^{(t)} \quad \Leftrightarrow \quad z_2^{(t)} + z_3^{(t)} - x_3^{(t)} \geq z_2^{(t)} \quad \Leftrightarrow \quad z_3^{(t)} \geq x_3^{(t)} ,$$

which is true. Therefore, $\boldsymbol{u}^{(t)} \triangleright \boldsymbol{z}^{(t)}$, and also $\boldsymbol{u}^{(t+1)} \triangleright \boldsymbol{z}^{(t+1)}$. Furthermore, $\boldsymbol{x}^{(t)} = \boldsymbol{y}^{(t)} + \boldsymbol{u}^{(t)}$. Since $y_2^{(t)} < y_1^{(t)} + y_0^{(t)}$, $u_3^{(t+1)} - u_3^{(t)}$ is equal to

$$\begin{aligned}
\frac{1}{\ell_1}(u_2^{(t)} - (u_1^{(t)} + u_0^{(t)})) &= \frac{1}{\ell_1}(x_2^{(t)} - y_2^{(t)} - (x_1^{(t)} - y_1^{(t)} + x_0^{(t)} - y_0^{(t)})) \\
&= \frac{1}{\ell_1}(x_2^{(t)} - (x_1^{(t)} + x_0^{(t)})) - \frac{1}{\ell_1}(y_2^{(t)} - (y_1^{(t)} + y_0^{(t)})) \geq x_3^{(t+1)} - x_3^{(t)}
\end{aligned}$$

Thus, $\boldsymbol{x}^{(t+1)} \triangleright \boldsymbol{y}^{(t+1)} + \boldsymbol{u}^{(t+1)} \triangleright \boldsymbol{y}^{(t+1)} + \boldsymbol{z}^{(t+1)}$. This completes the proof of the claim. $\square$

Furthermore, the next claim implies that $\boldsymbol{z}^{(t)} \triangleright \boldsymbol{c}^{(t)}$ for all $t \geq t_2 - 1$. (Recall that $\boldsymbol{z}^{(t)}$ and $\boldsymbol{c}^{(t)}$ start with the same distribution, but $\boldsymbol{c}^{(t)}$ is applied to p whereas $\boldsymbol{z}^{(t)}$ is applied to p'.)

Claim 6.49 *Consider the case that $(a_0^{(0)}, a_1^{(0)}, a_2^{(0)}, a_3^{(0)}) = (-\epsilon, \epsilon, 0, 0)$ and for the rest of the time steps nothing is moved from a_0 to a_1. Then we get with MAXDELAY that, for every $t \in \mathbb{N}$,*

$$\begin{aligned}
a_1^{(t)} &= \left(2\left(1 - \frac{1}{\ell_1}\right)^t - 1\right)\epsilon , \\
a_2^{(t)} &= 2\left[\left(\frac{1}{\ell_1} + \frac{1}{\ell_2}\right)\sum_{i=0}^{t-1}\left(1 - \frac{1}{\ell_1}\right)^i\left(1 - \frac{1}{\ell_2}\right)^{(t-1)-i} - \left(1 - \left(1 - \frac{1}{\ell_2}\right)^t\right)\right]\epsilon , \\
a_3^{(t)} &= 2\left[2 - \left(\frac{1}{\ell_1} + \frac{1}{\ell_2}\right)\sum_{i=0}^{t-1}\left(1 - \frac{1}{\ell_1}\right)^i\left(1 - \frac{1}{\ell_2}\right)^{(t-1)-i} - \left(1 - \frac{1}{\ell_1}\right)^t - \left(1 - \frac{1}{\ell_2}\right)^t\right]\epsilon .
\end{aligned}$$

Proof. We prove the formulas above by complete induction. Let us start with $a_1^{(t)}$. For $t = 0$ the formula is obviously correct. Suppose it is correct for some $t \geq 0$. Then it holds for step $t + 1$:

$$\begin{aligned}
a_1^{(t+1)} &= a_1^{(t)} - \tfrac{1}{\ell_1}(a_1^{(t)} - a_0^{(t)}) \\
&= \left(2\left(1 - \frac{1}{\ell_1}\right)^t - 1\right)\epsilon - \frac{1}{\ell_1}\left[\left(2\left(1 - \frac{1}{\ell_1}\right)^t - 1\right)\epsilon + \epsilon\right] \\
&= \left(2\left(1 - \frac{1}{\ell_1}\right)^t - 2\frac{1}{\ell_1}\left(1 - \frac{1}{\ell_1}\right)^t - 1\right)\epsilon \\
&= \left(2\left(1 - \frac{1}{\ell_1}\right)^{t+1} - 1\right)\epsilon .
\end{aligned}$$

Next we prove the formula for $a_2^{(t)}$. For $t=0$, it is clearly correct. Suppose the formula is correct for some $t \geq 0$. Then it holds for step $t+1$:

$$\begin{aligned}
a_2^{(t+1)} &= a_2^{(t)} + \tfrac{1}{\ell_1}(a_1^{(t)} - a_0^{(t)}) - \tfrac{1}{\ell_2}(a_2^{(t)} - a_1^{(t)} - a_0^{(t)}) \\
&= \left(1-\tfrac{1}{\ell_2}\right)a_2^{(t)} + \left(\tfrac{1}{\ell_1}+\tfrac{1}{\ell_2}\right)a_1^{(t)} + \left(\tfrac{1}{\ell_1}-\tfrac{1}{\ell_2}\right)\epsilon \\
&= \left(1-\frac{1}{\ell_2}\right)a_2^{(t)} + \left(\frac{1}{\ell_1}+\frac{1}{\ell_2}\right)\left(2\left(1-\frac{1}{\ell_1}\right)^t - 1\right)\epsilon + \left(\frac{1}{\ell_1}-\frac{1}{\ell_2}\right)\epsilon \\
&= a_2^{(t)} - \frac{2}{\ell_2}\left(\frac{1}{\ell_1}+\frac{1}{\ell_2}\right)\epsilon\sum_{i=0}^{t-1}\left(1-\frac{1}{\ell_1}\right)^i\left(1-\frac{1}{\ell_2}\right)^{(t-1)-i} - \frac{2}{\ell_2}\left(1-\frac{1}{\ell_2}\right)^t\epsilon + \\
&\quad 2\left(\frac{1}{\ell_1}+\frac{1}{\ell_2}\right)\left(1-\frac{1}{\ell_1}\right)^t\epsilon \\
&= 2\left(\frac{1}{\ell_1}+\frac{1}{\ell_2}\right)\left[\left(1-\frac{1}{\ell_2}\right)\sum_{i=0}^{t-1}\left(1-\frac{1}{\ell_1}\right)^i\left(1-\frac{1}{\ell_2}\right)^{(t-1)-i} + \left(1-\frac{1}{\ell_1}\right)^t\right]\epsilon - \\
&\quad 2\left(1-\left(1-\frac{1}{\ell_2}\right)\left(1-\frac{1}{\ell_2}\right)^t\right)\epsilon \\
&= 2\left[\left(\frac{1}{\ell_1}+\frac{1}{\ell_2}\right)\sum_{i=0}^{t}\left(1-\frac{1}{\ell_1}\right)^i\left(1-\frac{1}{\ell_2}\right)^{t-i} - \left(1-\left(1-\frac{1}{\ell_2}\right)^{t+1}\right)\right]\epsilon\,.
\end{aligned}$$

It remains to prove the formula for $a_3^{(t)}$. Since $\sum_{i=0}^{3} a_i^{(t)} = 0$ for all $t \geq 0$ we get:

$$\begin{aligned}
a_3^{(t)} &= \epsilon - a_2^{(t)} - a_1^{(t)} \\
&= \left[1 - 2\left[\left(\frac{1}{\ell_1}+\frac{1}{\ell_2}\right)\sum_{i=0}^{t-1}\left(1-\frac{1}{\ell_1}\right)^i\left(1-\frac{1}{\ell_2}\right)^{(t-1)-i} - \left(1-\left(1-\frac{1}{\ell_2}\right)^t\right)\right] - \right. \\
&\quad \left.\left(2\left(1-\frac{1}{\ell_1}\right)^t - 1\right)\right]\epsilon \\
&= 2\left[2 - \left(\frac{1}{\ell_1}+\frac{1}{\ell_2}\right)\sum_{i=0}^{t-1}\left(1-\frac{1}{\ell_1}\right)^i\left(1-\frac{1}{\ell_2}\right)^{(t-1)-i} - \left(1-\frac{1}{\ell_2}\right)^t - \left(1-\frac{1}{\ell_1}\right)^t\right]\epsilon\,.
\end{aligned}$$

This completes the proof of the claim. □

Since, for all $t \geq t_2$, $\boldsymbol{a}^{(t)} = \boldsymbol{b}^{(t)} + \boldsymbol{c}^{(t)}$ according to Claim 6.47, $\boldsymbol{x}^{(t)} \rhd \boldsymbol{y}^{(t)} + \boldsymbol{z}^{(t)}$ according to Claim 6.48 and $\boldsymbol{z}^{(t)} \rhd \boldsymbol{c}^{(t)}$ according to Claim 6.49, it holds that:

In order to show that $\boldsymbol{x}^{(t)} \rhd \boldsymbol{a}^{(t)}$ also for $t > t_2$ it remains to show that $\boldsymbol{y}^{(t)} \rhd \boldsymbol{b}^{(t)}$ for all $t > t_2$. Because $y_3^{(t_2+1)} = b_3^{(t_2+1)} = 0$, this is certainly true for $t = t_2+1$. Applying the same reduction to $\boldsymbol{b}^{(t)}$ and $\boldsymbol{y}^{(t)}$ for time step t_2+2 as done to $\boldsymbol{a}^{(t)}$ and $\boldsymbol{x}^{(t)}$ for time step t_2+1, proves that also $\boldsymbol{x}^{(t_2+2)} \rhd \boldsymbol{a}^{(t_2+2)}$. Continuing this for time steps $t_2+3, t_2+4, \ldots$ yields the proposition. □

From the proposition it immediately follows that dominance is preserved. This proves the lemma. □

Lemma 6.45 implies that we can order the edges in p in such a way that first all edges with latency L are used, then all edges with latency $L/2$ are used, and so on, until we arrive at the edges with latency 2. Let the resulting schedule be called $\mathcal{S}_4$. Let $\mathcal{S}_{4,i}$ denote the part of $\mathcal{S}$ that contains all edges of latency 2^i. According to the uniform case, for each one of the $\mathcal{S}_{4,i}$, it takes at most $O(T + 2^i \log n)$ steps, w.h.p., to complete the simulation of $\mathcal{S}_{4,i}$. Hence, the total amount of time steps required, w.h.p., to simulate $\mathcal{S}_4$ is bounded by

$$\sum_{i=\log \ell}^{\log L} O\left(T + 2^i \log n\right) = O(T \log(L/\ell) + L \log n) \ .$$

Hence, after $O(T \log(L/\ell) + L \log N)$ time steps the simulation of $\mathcal{S}_1$ in G is completed, w.h.p., which concludes the proof of Theorem 6.13.

6.5 Conclusion

In this chapter, we presented two approaches for the design of efficient routing schemes for ad hoc wireless networks: first, we considered the effect of restricting the class of routing strategies allowed, and then we restricted the class of allowed transmission graphs. Both restrictions we considered led to permutation routing strategies that are close to optimal for their respective classes.

The model we used in this chapter has two major restrictions: we only considered static ad hoc networks, and we only studied static routing problems. In reality, however, ad hoc networks may dynamically change, and packets may be continuously injected into the network. Hence, one may ask, whether certain models or scenarios can also be identified for dynamic ad hoc networks that allow the design of (nearly) optimal static routing protocols, and how continuous injections can be handled efficiently in (static or dynamic) ad hoc networks. A major problem for the design of efficient protocols for dynamic ad hoc networks is to maintain paths that allow one to enable the exchange of messages in a way that is significantly better than just using broadcasting.

Chapter 7

Job Shop Scheduling

In this chapter, we will present upper and lower bounds for job shop scheduling problems, concentrating on acyclic job shop scheduling. In acyclic job shop scheduling problems there are n jobs and m machines. Each job is composed of a sequence of operations to be performed on different machines. A legal schedule is one in which within each job, operations are carried out in order, and each machine performs at most one operation in any unit of time. If D denotes the length of the longest job, and C denotes the number of time units requested by all jobs on the most loaded machine, then clearly $lb = \max[C, D]$ is a lower bound on the length of the shortest legal schedule. A celebrated result of Leighton, Maggs, and Rao shows that if all operations are of unit length, then there always is a legal schedule of length $O(lb)$, independent of n and m. For the case that operations may have different lengths, Shmoys, Stein, and Wein showed that there always is a legal schedule of length $\tilde{O}(lb(\log lb)^2)$, where the $(\tilde{O})$ notation is used to suppress $\log\log(lb)$ terms. We improve the upper bound to $\tilde{O}(lb \log lb)$. We also show that our new upper bound is essentially best possible, by proving the existence of instances of acyclic job shop scheduling for which the shortest legal schedule is of length $\tilde{\Omega}(lb \log lb)$. This resolves (negatively) a known open problem of whether the linear upper bound of Leighton, Maggs, and Rao applies to arbitrary job shop scheduling instances (without the restriction to acyclicity and unit length operations).

7.1 Introduction

In job shop scheduling (JSS) problems there are n jobs and m machines. Each job is composed of a sequence of operations, where each operation has a machine and length (processing time) associated with it. An operation is scheduled by specifying a time step at which it begins, and then the operation is executed on its respective machine without interruption for a time period that is equal to its length. A legal schedule is one in which each operation is scheduled only after all operations preceding it in its job have been completed, and each machine performs at most one operation at any unit of time. The makespan of a legal schedule is the time by which all jobs have been completed. For a given JSS instance, let L denote the minimum makespan, taken over all possible legal schedules. Computing L is NP-hard. In this chapter, we are interested in providing upper and lower bounds on L.

We shall concentrate on acyclic job shop scheduling, for which within each job, all operations are performed on different machines. A known example where acyclic JSS comes up naturally is that of message routing through a communication network. Each link of the network may be viewed as a machine, and each message may be viewed as a job. If a path is specified from the source to the destination of a message, traversing a link along the path can be viewed as an operation. Naturally, paths are acyclic. How should one schedule the transmissions of messages over links so as to minimize the time until all messages reach their destinations? If messages cannot be segmented and need to hop from node to node as a whole, and furthermore, while a message traverses a link no other message can use this link, then this question is an instance of acyclic JSS.

There are two trivial lower bounds on L. One of them is the length of the longest job, known as the dilation D. The other is the load on the most loaded machine, namely, the total number of time units that all jobs require on this machine, which is known as the congestion C. Clearly, $lb = \max[C, D]$ is a lower bound on L. In a seminal paper, Leighton, Maggs, and Rao [LMR94] showed that for acyclic JSS, if all operations are of unit length, then in fact $L = \Theta(lb)$, regardless of any other parameter such as n and m. The proof given in [LMR94] is non-constructive (in a sense that it does not provide an efficient algorithm for finding a schedule with makespan $O(lb)$) and makes repeated use of the Lovász Local Lemma. Attempts to generalize the $O(lb)$ upper bound to general JSS (with the assumptions on acyclicity or unit length removed) were unsuccessful. The best upper bounds known for general JSS are $L = O(lb(\log lb)^2/\log\log lb)$ by Shmoys, Stein, and Wein [SSW94], later improved by a $\log\log lb$ factor by Goldberg, Paterson, Srinivasan, and Sweedyk [GPSS97].

Our main results are as follows:

- For acyclic JSS, $L = O(lb \log lb \log\log lb)$. This improves previously known upper bounds for acyclic JSS by a factor of $\log lb/(\log\log lb)^3$.
- There are instances of acyclic JSS for which $L = \Omega(lb \log lb/\log\log lb)$. This resolves (negatively) the open question (see, e.g., [SSW94, GPSS97]) of whether the linear upper bound of Leighton, Maggs, and Rao applies to arbitrary job shop scheduling instances. Moreover, this shows that our upper bound for acyclic JSS is best possible up to a factor of $O((\log\log lb)^2)$.

Similar to the proof in [LMR94], the proof of our upper bound makes repeated use of the Lovász Local Lemma. However, the existence of operations of different lengths requires the use of the general form of the Local Lemma, rather than the symmetric form used in [LMR94].

Our upper bound only proves the existence of short schedules. It does not provide an efficient algorithm for finding them. The main obstacle is the non-constructive nature of the Local Lemma. As shown in Section 3.6.2, there are algorithmic forms of the Local Lemma, and they have been used, for instance, to provide a polynomial time algorithm for finding schedules of length $O(lb)$ for unit length acyclic JSS [LMR99]. Providing an algorithmic form of our upper bound is much more complicated and is therefore deferred to Chapter 9, after some preparations in Chapter 8. We remark that for general JSS, the upper bounds of [SSW94, GPSS97] are algorithmic, and that it is NP-hard to approximate L within a ratio better than 5/4 [WHH$^+$96] (even for flow shop scheduling, which is a special case of acyclic JSS). In the context of packet switching, one often seeks an algorithm that is not only polynomial time, but also distributed. Some work towards providing a distributed algorithmic form of [LMR94]'s schedule is presented in [LMR94, RT96, OR97].

We point out some additional consequences of our work.

- **Preemption provably helps:** In preemptive JSS one is allowed to temporarily suspend (interrupt) an operation on some machine, let the machine perform some other operation, and later resume the original operation (as if an operation of length l is composed of a sequence of l unit length operations). In some cases of JSS it makes sense to allow preemption, perhaps at some cost. Previous to our work, there was no proof that using preemption can significantly reduce the makespan. In Section 7.3.1 we show an example where this is indeed the case. More importantly, we are also able to prove a general upper bound of $L = O(lb \log\log lb)$ for acyclic JSS with preemption. Note that a similar upper bound does not hold if preemption is not allowed, by our lower bound of $L = \Omega(lb \log lb / \log\log lb)$.

- **Dilation versus congestion:** It has been observed that congestion and dilation influence the makespan differently. For example, Leighton, Maggs, and Rao [LMR88] present a distributed algorithm for unit length acyclic JSS that produces schedules of length $O(C + D\log n)$. Our results (see Sections 7.2 and 7.3 for details) make similar distinctions between C and D. In particular, if $C = \Omega(D^{1+\epsilon})$ for some $\epsilon > 0$, then $L = \Theta(lb)$, even for general JSS.

- **Operation lengths that depend only on machine:** In a packet routing network, it may well be the case that all packets are of the same size, but that some links are slower than others (modems of different speeds on the communication lines). In this case, any two packets take the same time to cross the same link, but a packet may take more time to cross one link than another link. For acyclic JSS in which operation lengths depend only on the machine, our upper bound improves to $L = O(lb \log\log lb)$. (For general JSS, if operation lengths depend only on the machine, then $L = O(lb \log lb / \log\log lb)$. This follows from techniques of [LMR94, SSW94].)

- **Operation lengths that depend only on job:** Returning to the routing example, consider the case where each link has the same speed, but messages may be of different length. In this case, different messages may take different time to cross the same link, but any message takes the same time to cross two different links. This restriction of acyclic JSS does not seem to help, as in our negative example for which $L = \Omega(lb \log lb / \log\log lb)$, operation lengths depend only on the job. (Any instance of acyclic JSS can also be cast as an instance of message routing, though the paths that messages follow from source to destination may appear to be artificial.) This finding supports the practice of breaking messages into packets. By this we obtain an instance of unit length acyclic JSS, which by [LMR94] has a makespan of $O(lb)$. We note that we may use here a convenient form of packet routing in which all packets corresponding to the same message follow the same path and arrive in order. If, however, we add the requirement that the packets of a message have to form a continuous stream through the network, Cole, Maggs and Sitaraman [CMS96] showed that under certain circumstances the makespan may again be in $\omega(lb)$.

- **A constant number of operation lengths:** If there are only a constant number of different operation lengths, then we can show that $L = O(lb)$ (the O notation hides a constant that is exponential in the number of different operation lengths).

7.1.1 New Results

We assume that the shortest operation length is 1 (this is achieved by scaling), and that all other operation lengths are a power of 2 (by rounding operation lengths up to the nearest power of 2, loosing a factor of at most 2 in congestion and dilation). Recall that C and D denote congestion and dilation (respectively), and that $lb = \max[C, D]$. Let P denote the length of the longest operation, and for general JSS, let Q denote the maximum number of time units that one job requests one machine (i.e., the sum of the lengths of the operations that the job performs on that machine). Hence $P \leq Q \leq \min[C, D] \leq lb$. In the following, we assume that if $\log x$ or $\log\log x$ are smaller than 1, we always round them up to 1.

The following two theorems, which are proved in Section 7.2, describe our upper bounds on the length L of the shortest legal schedule.

Theorem 7.1 *For acyclic job shop scheduling, the following upper bounds on the makespan hold:*

(1) For any acyclic JSS problem,

$$L = O\left((C + D\log\log P)\frac{\log P}{\log(\min[C/D + \log\log P, P])}\right),$$

implying that $L = O(lb \log lb \log\log lb)$. Observe that if $C \geq D \cdot P^{\epsilon}$ for some $\epsilon > 0$, then $L = \Theta(C)$.

(2) If there are only a constant number of operation lengths (independent of the values of C and D), then $L = \Theta(lb)$.

(3) If operation lengths depend only on the machine on which the operation is performed, then $L = O(C + D\log\log P)$.

(4) With preemption, $L = O(C + D\log\log P)$.

Theorem 7.2 *For general job shop scheduling, the following upper bounds on the makespan hold:*

(1) For any JSS problem,

$$L = O\left(\left(C + D\frac{\log Q}{\log\left(2 + \frac{\log Q}{1+C/D}\right)}\right)\frac{\log P}{\log(\min[C/D + \log Q, P])}\right).$$

Observe that if $C \geq D\max[P^{\epsilon}, \log Q]$ for some $\epsilon > 0$, then $L = \Theta(C)$.

(2) With preemption, $L = O(C + (D\log Q)/\log(2 + \frac{\log Q}{1+C/D}))$.

The next two theorems, which are proved in Section 7.3, describe our lower bounds on L. For general JSS, we give a simple explicit construction showing:

Theorem 7.3 *For any integer $k > 1$, there is an instance of (general) JSS with k jobs, k machines, dilation and congestion each equal to $D = (k\log k)^k$, such that any legal schedule is of length more than $Dk(1 - \frac{1}{\log k})$. For large k this gives $L \simeq kD$ and $k \simeq \log D/\log\log D$.*

Using a probabilistic argument, the following result is shown for acyclic JSS.

Theorem 7.4 *For a large enough dilation D, there is an instance of acyclic JSS with $C < D$ and $L > D\frac{\log D}{16\log\log D}$.*

7.2 Upper Bounds

In this section we prove Theorems 7.1 and 7.2. Let us first give a high level overview of our approach and the new ingredients that it contains compared to [LMR94, SSW94, GPSS97]. For simplicity, consider an instance of acyclic job shop scheduling with $n = m = D = C$ and operations of varying lengths. The approach of [SSW94] (based on techniques used in [LMR94]) for handling this instance has two phases:

(1) Use a simple randomized algorithm to construct an intermediate schedule of length $O(n)$. This intermediate schedule may be illegal in the sense that it contains contention conflicts: at certain time steps as many as $\alpha = O(\log n / \log\log n)$ jobs may want to operate on the same machine.

(2) Convert the intermediate schedule into a legal schedule of length $O(\alpha n \log n)$.

In [GPSS97], the second phase was improved to give a schedule of length $O(\alpha n \log n / \log\log n)$.

We also follow this two phase approach. In the intermediate schedule we allow for higher contention $\alpha = O(\log n \log\log n)$. However, this contention is distributed evenly over operations of different lengths, where for each value of t, operations of length t contribute at most $O(\log\log n)$ to the contention. This allows for a more efficient implementation of the second phase, which now gives a legal schedule of length essentially $O(\alpha n)$.

In order to construct our intermediate schedule, we follow the approach of [LMR94]. We start with the (illegal) schedule of running all jobs in parallel and make a sequence of refinement steps, each time obtaining a new (illegal) schedule. The feasibility of each refinement step is proved using the Lovász Local Lemma. However, the existence of operations of different lengths (rather than just unit length operations) causes significant differences between our refinement steps and those of [LMR94]. We use many more refinement steps ($O(\log n / \log\log n)$ rather than $O(\log^* n)$), and we use the general form of the Local Lemma rather than the uniform (symmetric) form.

Section 7.2.1 contains some definitions and establishes some notation that is used throughout the proofs of the upper bounds. In Section 7.2.2, we show how to convert suitable intermediate schedules for any JSS problem into legal schedules. Section 7.2.3 proves the existence of an efficient intermediate schedule for general JSS (used in the proof of Theorem 7.2), and Section 7.2.4 proves the existence of an efficient intermediate schedule for acyclic JSS (used in the proof of items (1), (3), and (4) in Theorem 7.1). In Section 7.2.5, we present the proof of Theorem 7.1 (2).

7.2.1 Definitions

Our proofs will use the following notation. A t-operation is an operation of length t. We assume that the shortest operation length is 1 (this is achieved by scaling), and that all other operation lengths are a power of 2 (by rounding operation lengths up to the nearest power of 2, loosing a factor of at most 2 in congestion and dilation). We use the notation $(\leq t)$-operation to denote an operation whose length is not more than t, and T-operation to denote an operation whose length belongs to the set T.

A time interval is composed of consecutive time units. A *t-interval* denotes a time interval of length t. Some intervals, called *frames*, are special. For a frame F, its length $|F|$ is always a power of two, and its starting time is an integer multiple of $|F|$. A schedule is called *well-structured* if every t-operation falls into a frame of length t.

For a (possibly illegal) schedule, the $(\leq t)$-*congestion* for machine M in time interval I is the sum over all $(\leq t)$-operations that are scheduled to begin execution on M within I, where each such t'-operation contributes t' to the sum. The T-*contention* of machine M is the maximum number of T-operations that are scheduled on it at one time unit (for a legal schedule, the contention of every machine is one).

As the length of the largest operation is P, there are at most $1 + \log P$ different operation lengths. Partition operation lengths into consecutive sets of cardinality c. These sets are denoted in the following by $T_1, T_2, \ldots, T_{(1+\log P)/c}$, where set T_i contains all operation lengths from $2^{(i-1)c}$ to 2^{ic-1}. (For acyclic JSS, we shall later fix $c = \log\log P$.) We remark that in Section 7.2.4 we shall use a slightly modified definition for T_i. This modified definition appears in the beginning of Section 7.2.4.

7.2.2 Transforming Illegal into Legal Job Shop Schedules

The following lemma describes an intermediate schedule that may be illegal (in the sense that a machine may need to perform several operations at the same time), but can be transformed into a legal schedule with relatively short makespan. The lemma holds for general JSS (acyclicity is not required).

Lemma 7.5 *Let $\alpha, \beta, \gamma, \delta$ be nonnegative parameters that may depend on C and D. Assume a well-structured schedule of length δD with the following properties holding for every machine:*

- *For every set T_i, the T_i-contention is at most $\alpha + \beta C/D$.*
- *For any frame F, where $|F|$ is a power of 2^c, the $(< |F|/2^c)$-congestion is at most $\gamma|F|C/D$.*

Then the intermediate schedule can be transformed into

(1) a legal schedule of length $L \leq 2\delta D\phi \log P/\log(\min[\phi, P])$ with $\phi = 2\alpha + (2\beta + \gamma)C/D$ for the general case,

(2) a legal schedule of length $L \leq \delta(\alpha D + \beta C)$ for the case that the operation length depends only on the machine on which the operation is performed, or

(3) a preemptive schedule of length $L \leq 2\delta(\alpha D + (\beta + \gamma)C)$.

Assume that Lemma 7.5 is true. Then Theorem 7.1 (1), (3), and (4) would follow if we could prove the existence of a well-structured schedule for acyclic JSS with

$$c = \log\log D, \quad \alpha = O(\log\log P), \quad \gamma = O(1 + D/C), \quad \text{and} \quad \beta, \delta = O(1) .$$

Theorem 7.2 would follow if we could prove the existence of a well-structured schedule for general JSS with

$$c = \log D, \quad \alpha = O\left(\frac{\log Q}{\log\left(2 + \frac{\log Q}{C/D+1}\right)}\right), \quad \gamma = O(1 + D/C), \quad \text{and} \quad \beta, \delta = O(1) .$$

Let us first prove Lemma 7.5. In the following, let $\mathcal{S}$ denote a schedule that fulfills the requirements of Lemma 7.5.

Proof of Item 1

The proof extends a technique by Goldberg et al. [GPSS97]. Let us start with partitioning $\mathcal{S}$ into frames of length P. By the definition of P and the fact that $\mathcal{S}$ is well-structured, no operation crosses a frame border. We will show how to construct a legal schedule for the operations in each frame. Concatenating these schedules yields a legal schedule for the whole JSS problem.

Consider some fixed frame F. Let T be a rooted complete binary tree with P leaves labeled, from left to right, $0, 1, \ldots, P-1$, each leaf denoting a time step of the frame. Let u be a node in T and let $\ell(u)$ and $r(u)$ be the labels of the leftmost and rightmost leaves of the subtree rooted at u. For every machine M_i, we define $S_i(u)$ to be those operations that are scheduled on M_i in F for *precisely* the time interval $[\ell(u), r(u)]$. Hence each operation is in exactly one $S_i(u)$. In the following, we will describe a frame scheduling algorithm that produces a legal schedule for F with makespan $2\phi P \frac{\log P}{\log(\min[\phi,P])}$, where $\phi = 2\alpha + (2\beta+\gamma)C/D$.

Let us assume in the following that ϕ is a power of two. Mark a node u if it is at height $(0 \bmod \log\phi)$ in T. (If u is a leaf, then its height is 0, the father of u has height 1, and so on.) Eliminating the edges between a marked node and its children partitions T into a collection of subtrees, each of height (at most) $\log\phi$.

Let T' be one of the subtrees of the partition. We move all operations stored in T' from the sets $S_i(v)$ to new sets $S'_i(v)$ for every machine M_i. Initially, each $S'_i(u)$ is empty for all $u \in T'$. We describe a procedure for building up the sets $S'_i(u)$. At each step of this procedure, $p_i(u)$ denotes the time needed to process all operations currently in $S'_i(u)$. We scan the nodes of T' from the bottom upwards, starting with all nodes at height 0, then continuing with all nodes at height 1, and so on until the root of T' is reached. For a node v in T' we scan its operations in an arbitrary order. For each such operation that has to be performed on machine M_i (i.e., this is an operation belonging to $S_i(v)$) we place it in $S'_i(w)$, where w is a leaf of the subtree of T' rooted at v with the currently smallest $p_i(w)$. We update the respective $S'_i(w)$ and $p_i(w)$. In essence, this procedure amounts to:

Distribute all operations of $S_i(v)$ among the $S'_i(w)$ of its leaves w such that their $p_i(w)$'s are as balanced as possible.

We now view T once again as one complete binary tree. For each vertex u we define $p(u) = \max_i p_i(u)$. Let the nodes of T be numbered as $u_1, u_2, \ldots$ in the preorder traversal of T. Define $f(u_1) = 0$, and for any $j \geq 2$ let $f(u_j) = \sum_{k<j} p(u_k)$. Run the following scheduling algorithm to produce a schedule $\mathcal{S}_F$:

Schedule the operations in $S'_i(u)$ on machine M_i consecutively beginning at time step $f(u)$ and concluding before $f(u) + p(u)$.

It remains to show that $\mathcal{S}_F$ is legal and has a makespan of at most $2\phi P \frac{\log P}{\log(\min[\phi,P])}$. Consider any subtree T' of the partition. Assume the leaves of T' are at height j in T. Clearly, the algorithm for distributing the operations in the $S_i(u)$ among the $S'_i(u)$ ensures that all $S'_i(u)$ for all nodes in T' are empty except for the leaves of T'. We want to show by complete induction that for every height $h \in \{0, \ldots, \log\phi\}$, after the algorithm has distributed all operations in the S_i of nodes of height h in T' among the S'_i of their leaves, $p(u) \leq \phi 2^j + 2^{j+h}$ for every leaf u in T'.

For $h = 0$, this assumption follows directly from the contention bound in Lemma 7.5. Now suppose that the bound on the $p(u)$'s is true for some $h \geq 0$. Then we want to show that it also holds for $h + 1$.

Assume in the contrary that the assumption is not true for $h + 1$. Then there must exist a machine M_i and a node v at height $h + 1$ in T' with one of its leaves u having $p_i(u) > \phi 2^j + 2^{j+h+1}$. Since after the distribution of operations in nodes of height h, $p_i(u)$ still was at most $\phi 2^j + 2^{j+h}$, u must have an operation from $S_i(v)$. If this operation cannot be moved to another leaf to decrease the number of leaves w with $p_i(w) > \phi 2^j + 2^{j+h+1}$, $p_i(w)$ must be greater than $\phi 2^j$ for all leaves w under v. This, however, induces a total congestion of more than $2^{h+1} \cdot \phi 2^j$ by the $(\leq 2^{j+h+1})$-operations stored in the subtree of v, which is a contradiction to the contention and congestion bounds of Lemma 7.5 (two times the contention bound for the largest operations plus the congestion bound for the rest are sufficient to bound the total congestion in the subtree of v). Hence, the operations can be distributed in such a way that for all leaves u it holds

$$p_i(u) \leq \phi 2^j + 2^{j+\log \phi} = \phi 2^{j+1} .$$

There are $P/2^j$ nodes at this height in T. The sum of these $p(u)$'s is thus at most $2\phi P$. Each subtree layer therefore contributes at most $2\phi P$, and there are $\frac{\log P}{\log(\min[\phi,P])}$ layers. Thus, $\sum_{v \in T} p(v)$, the makespan of schedule $\mathcal{S}_F$, is at most $2\phi P \frac{\log P}{\log(\min[\phi,P])}$.

To show that $\mathcal{S}_F$ is legal we observe that by construction, no machine performs more than one operation at a time. We also need to show that each job performs its operations in order. For this, consider two operations O_1 and O_2 of the same job, where O_2 follows O_1. Then we have that O_1 finishes before O_2 begins under $\mathcal{S}$. Assume (the more difficult case) that $\mathcal{S}$ schedules O_1 and O_2 within the same frame F. Consider the respective tree T described above and let nodes u and v be such that $O_1 \in S_i(u)$, $O_2 \in S_j(v)$. Then u and v are roots of disjoint subtrees of T and u precedes v in the preorder traversal of T. Thus O_1 finishes before O_2 begins in $\mathcal{S}_F$, and the new schedule is feasible.

Concatenating all $\delta D/P$ legal frame schedules yields a legal schedule whose makespan is bounded as described in Lemma 7.5(1).

Proof of Item 2

For item 2 (operation length depends on machine), observe first that the contention is an integer, and hence $\alpha + \beta C/D$ is an integer. Modify the original schedule in a greedy way such that each l-operation on machine M that started at time t (recall that t is an integer multiple of l) now starts at a time step $(\alpha + \beta C/D)t + il$, where $0 \leq i < (\alpha + \beta C/D)$, and i is chosen such that no previously considered l-operation on machine M is scheduled to begin at the same time. Clearly, the length of the new schedule is at most $\delta D(\alpha + \beta C/D)$. The new schedule is legal for the following reasons. For each job, operations are performed in order, because they were performed in order in schedule $\mathcal{S}$, and every operation that started at time t_1 and ended at time t_2 in $\mathcal{S}$ now starts at time $t_1' \geq (\alpha + \beta C/D)t_1$ and ends at time $t_2' \leq (\alpha + \beta C/D)t_2$ in the new schedule. Furthermore, each machine performs at most one operation at every unit of time, because all operations in one machine are of the same length.

Proof of Item 3

Let us divide all operations into the following two sets:

- Set A consists of all T_i-operations, where $i \in \{1, \ldots, \frac{1+\log P}{c}\}$ is odd.
- Set B consists of all T_i-operations, where $i \in \{2, \ldots, \frac{1+\log P}{c}\}$ is even.

We first show that for each of the sets it is possible to produce a (still illegal) schedule $\mathcal{S}'$ with contention $\alpha + (\beta + \gamma)C/D$ such that no operation starts earlier than supposed to start according to $\mathcal{S}$ and ends later than supposed to end according to $\mathcal{S}$. We achieve this by suitably distributing the units of processing among the time steps, allowing several units of processing from the same operation to be placed at the same time step.

W.l.o.g., let us concentrate on scheduling operations in A. First, let us schedule the T_1-operations. According to the contention bound in Lemma 7.5, this can be done with contention $\alpha + \beta C/D$ without moving the operations.

Suppose now that for some odd $i \in \{3, \ldots, \frac{1+\log P}{c}\}$ all T_j-operations in A with $j < i$ can be scheduled with contention $\alpha + (\beta + \gamma)C/D$. Then we want to show that also the T_j-operations in A with $j \leq i$ can be scheduled with contention $\alpha+(\beta+\gamma)C/D$. Let us choose the strategy that we always try to fully exploit this contention bound, preferring insertions of units of processing from the operation that has to finish next in $\mathcal{S}$. Suppose that with this strategy there is an operation that cannot be inserted before it ends according to $\mathcal{S}$. We will show by contradiction that this cannot happen.

Let us choose the first operation, say O, where a bad event happens. Let us go back in time until we reach a point where a contention of $\alpha + (\beta + \gamma)C/D$ could not be fully exploited (or we reach the beginning of $\mathcal{S}$). This means that all T_i-operations that started before could already be inserted. Choose I as the interval that starts at the first point (forward in time) from this point, where a T_i-operation starts in $\mathcal{S}$, and ends where O is supposed to end in $\mathcal{S}$. Since the starting point of I cannot be later than the time step when O starts in $\mathcal{S}$, the length of I is at least $2^{(i-1)c}$. In fact, we chose I above such that its length is an integer multiple of $2^{(i-1)c}$ (which is the smallest length a T_i-operation can have). Since $\mathcal{S}$ is well-structured, all units of processing of T_j-operations with $j < i$ that are inserted in I belong to T_j-operations that start in I. From the congestion bound in Lemma 7.5 we know that within such an interval the congestion caused by T_j-operations in A with $j < i$ is at most $\gamma|I|\frac{C}{D}$. Hence, if a contention of $\alpha + (\beta + \gamma)C/D$ does not suffice to insert all T_i-operations that end no later than O, then, since all T_i-operations starting before I could be inserted before I, this means that somewhere in I there must have been a contention among the T_i-operations of more than $\alpha+\beta C/D$, which contradicts the contention bound in Lemma 7.5.

Hence, all operations in A can be scheduled with contention $\alpha+(\beta+\gamma)C/D$, and therefore all operations in A and B can be scheduled with contention $2(\alpha + (\beta + \gamma)C/D)$. Since a schedule of length t that has contention c can be simulated by a preemptive schedule of length $c \cdot t$ with contention 1, item 3 follows.

7.2.3 An Intermediate Schedule for General JSS

In this section, we prove the existence of an intermediate schedule that fulfills the requirements of Lemma 7.5 with

$$c = \log D, \quad \alpha = O\left(\frac{\log Q}{\log\left(2+\frac{\log Q}{1+C/D}\right)}\right), \quad \gamma = O(1 + D/C), \quad \text{and} \quad \beta, \delta = O(1)\ .$$

We assume w.l.o.g. that $C \geq D$. (If $D > C$, just use D instead of C as an upper bound on the congestion.) Since $c = \log D$, all operations (except of the D-operations) are combined into one set T_i. Hence, it remains to show the following lemma.

Lemma 7.6 *For any job shop scheduling problem, there exists a well-structured schedule of length $O(D)$ such that for every machine the contention is bounded by*

$$O\left(\frac{C}{D} + \frac{\log Q}{\log\left(2 + \frac{\log Q}{1+C/D}\right)}\right)\ .$$

Proof. Let us first assume that $Q \geq D^{1/4}$, i.e., $\log Q = \Theta(\log D)$. Consider, for each job, the random experiment of choosing a delay independently and uniformly at random from a range of $[0, D-1]$. Each job that is assigned a delay of δ waits for δ time steps and then is processed by the machines in the prescribed order until it is completed. We use this random experiment to show with the help of the Lovász Local Lemma that delays exist such that Lemma 7.6 is satisfied. In order to simplify the proof, let us assume in the following that D is large enough (in fact, $D \geq 8$ suffices).

Fix a machine M_i. We want to bound the probability that the contention among the operations in M_i exceeds some limit. For this, let us consider some fixed time step t. Let q be the probability that at least k units of processing are scheduled on M_i at t. There are at most $\binom{C}{k}$ ways to choose k units of processing from the operations for M_i. For each such choice, the probability that all k units are scheduled at t is at most $(1/D)^k$. Hence we get with $k = 2\mathrm{e}\frac{C}{D} + 4\log(CD)/\log(2 + \frac{\log(CD)}{1+C/D})$:

$$
\begin{aligned}
q &\leq \binom{C}{k}\left(\frac{1}{D}\right)^k \leq \left(\frac{\mathrm{e}C}{k\cdot D}\right)^k = \left(\frac{1}{2}\right)^{k\log\left(\frac{kD}{\mathrm{e}C}\right)} \\
&\leq \left(\frac{1}{2}\right)^{\frac{4\log(CD)}{\log\left(2+\frac{\log(CD)}{1+C/D}\right)}\cdot\log\left(2+\frac{\log(CD)}{C/D}\cdot\frac{1}{\log\left(2+\frac{\log(CD)}{1+C/D}\right)}\right)} \\
&\leq \left(\frac{1}{2}\right)^{2\log(CD)} = \frac{1}{(CD)^2}\ .
\end{aligned}
$$

Let us define event A_i to be true if M_i has a contention of at least k at some time step. Since there are at most $2D$ time steps to consider, the probability p that A_i is true is at most $1/(4CD)$.

In order to apply the Lovász Local Lemma, we have to bound the dependencies among all these events for all machines. Whether or not an event becomes true depends solely on the delays assigned to the jobs. Thus, two events are independent unless some job has operations in both of the corresponding machines. Since every machine processes at most C operations and each of these operations belongs to a job with at most D other operations, the dependence b among the events is at most $C \cdot D$. Hence, $\mathrm{e}p(b+1) < 1$ and therefore, by the Lovász Local Lemma, there exist delays such that the contention at every machine is bounded by

$$
2\mathrm{e}\frac{C}{D} + \frac{4\log(CD)}{\log\left(2+\frac{\log(CD)}{1+C/D}\right)} = O\left(\frac{C}{D} + \frac{\log D}{\log\left(2+\frac{\log D}{1+C/D}\right)}\right)\ .
$$

It remains to transform this schedule into a well-structured schedule. As it was already shown in [GPSS97], a non-well-structured schedule can be easily transformed into a well-structured schedule by stretching the makespan by a factor of two as follows:

For any l and t, operations of length l that start at time t in the original schedule start at a time step that is the least integer multiple of l that is at least as large as $2t$. Two operations overlap in the new schedule only if they overlap in the original schedule.

So for $Q \geq D^{1/4}$ the lemma is true. It remains to show what to do if $Q < D^{1/4}$.

Our strategy will be to make a succession of refinements to an initial schedule S_0 until we reach a schedule S_i of length $O(D)$ with the property that for every frame of size Q^3 the congestion at any machine is bounded by $O(\frac{C}{D}\cdot Q^3)$. Then we split S_i into frames of size Q^3 and schedule each of them independently as described above. This yields a final subschedule of length $O(Q^3)$ with contention

$$O\left(\frac{C}{D} + \frac{\log Q^3}{\log\left(2 + \frac{\log Q^3}{1+C/D}\right)}\right) .$$

Combining these subschedules into one schedule by executing them one after the other would therefore yield Lemma 7.6. In order to show how to do the refinements, we need the following proposition.

Proposition 7.7 *For any job shop scheduling problem with congestion C, dilation D, and $Q < D^{1/4}$, there exists a schedule of length $O(D)$ such that for any Q^3-interval the congestion at any machine is bounded by $O(\frac{C}{D} \cdot Q^3)$.*

Proof. The proof uses the Hoeffding bounds and the Lovász Local Lemma at each refinement step. Let us start with an initial schedule S_0, in which each job is executed at every time step until it is completed. This initial schedule is as short as possible; its length is only D. Let $I_0 = D$ and $I_j = \max[\log I_{j-1}, Q]$ for all $j \geq 1$. Our aim is to show that for all $j \geq 1$ with $I_j \geq \max[\log I_{j-1}, Q, 36]$ there exists a refinement from schedule S_{j-1} to S_j such that the congestion at any machine for any I_j^3-interval is bounded by $O(\frac{C}{D} \cdot I_j^3)$. So when $I_j = Q$ or $\log I_j < 36$ for the first time, we reached the situation claimed in the lemma. (The condition $I_j \geq 36$ was chosen to simplify the presentation of the proof. Basically, our techniques can be used for any $I_j \geq Q \geq 1$.)

The first step is to assign an initial delay to each job, chosen independently and uniformly at random from the range $[0, \Delta_1 - 1]$, where $\Delta_1 = D$. In the resulting schedule, S_1, a job that is assigned a delay of δ waits for δ steps and afterwards is processed without waiting again until it is completed. The length of S_1 is at most $2D$. We use the Lovász Local Lemma to show that if the delays are chosen independently and uniformly at random and I_1 is sufficiently large, then with nonzero probability the congestion at any machine in any I_1^3-interval is at most $C_1 = \frac{C}{D}(1 + \frac{4}{\sqrt{I_1}})I_1^3$. Thus, such a set of delays must exist.

To apply the Local Lemma, we associate a bad event with each machine. For machine M and time interval I, we define the congestion of M at I to be the sum of the lengths of those operations on M that start within time interval I. (Hence, operations that start in an earlier time interval and extend into I are not counted, whereas for operations that start in I we also count their part that extends into subsequent intervals.) The bad event for machine M is that there is some I_1^3-interval with a congestion of more than C_1. We bound the dependence b among the bad events and the probability p that a bad event occurs.

We first bound the dependence. Whether or not a bad event occurs depends solely on the delays assigned to the jobs that have operations for the corresponding machine. Thus, two bad events are independent unless some job has operations in both of the corresponding sets. Since each machine processes at most C operations and each of the corresponding jobs has operations in at most D other sets, the dependence b of the bad events is at most $C \cdot D$.

Next we bound the probability that a bad event occurs. Consider some fixed machine M and I_1^3-interval I. Let m be the number of operations for M. For every $i \in \{1, \ldots, m\}$, let the random variable X_i be the length of operation i if operation i is started during I and 0 otherwise. Let $X = \sum_{i=1}^m X_i$. Since the jobs choose their delays uniformly at random from a range of size Δ_1, we get $\Pr[X_i \neq 0] \leq I_1^3/\Delta_1$ for all $i \in \{1, \ldots, m\}$. Hence,

$$\mathrm{E}[X] = \sum_{i=1}^{m} \mathrm{E}[X_i] \leq \frac{C}{D} \cdot I_1^3 .$$

As every job has operations for at most Q time steps at M, the X_i's can be grouped together to independent random variables of value at most Q. Thus, we get together with the Hoeffding bounds and $\epsilon = \frac{4}{\sqrt{I_1}}$ (note that $\epsilon < 1$ because $I_1 \geq 36$) that

$$\Pr[X \geq C_1] = \Pr\left[X \geq (1+\epsilon)\tfrac{C}{D} \cdot I_1^3\right] \leq \mathrm{e}^{-\left(\frac{4}{\sqrt{I_1}}\right)^2 \frac{C}{D} \cdot I_1^3/3Q} \leq \mathrm{e}^{-4\frac{C}{D} \cdot I_1} \leq \frac{1}{(CD)^2} .$$

For the last inequality we used $I_1 \geq \log D$, and $C/D \geq \log C/\log D$ for $C \geq D > 2$. Since for each set there are at most $D + \Delta_1 = 2D$ many I_1^3-intervals to consider, the probability that a bad event occurs for machine M is bounded by $p \leq 1/(4CD)$. Thus, the product $\mathrm{e}p(b+1)$ is less than 1 and therefore, by the Lovász Local Lemma, there is an assignment of delays such that the congestion at every machine in every I_1^3-interval is bounded by C_1.

We now break schedule S_1 into frames of length I_1^4. With each frame we associate those operations that S_1 schedules to start within the time interval of the frame. For each job that has at least one operation associated with the frame, we consider the fragment of the job composed of those operations associated with the frame. For each frame separately, for each fragment of a job, we choose an additional delay relative to the starting point of the frame. Schedule S_2 then results from executing the modified frames one after the other.

To simplify terminology, we treat each fragment of a job as a job by itself. The final schedule will guarantee that fragments of the same job are scheduled in order. Consider now some fixed I_1^4-frame F in S_1. Let the delays of the jobs in F be chosen in the range $[0, \Delta_2 - 1]$, where $\Delta_2 = I_1^3 - I_2^3$. Hence, the length of the resulting schedule for this frame is at most $I_1^4 + I_1^3 + P \leq (1 + 2/I_1)I_1^4$ (recall that $P \leq Q \leq I_1$). We use the Lovász Local Lemma to show that if the delays are chosen independently and uniformly at random and I_2 is sufficiently large, then with nonzero probability the congestion in any I_2^3-interval is at most

$$C_2 = \left(1 + \frac{6}{\sqrt{I_2}}\right)\left(\frac{C_1}{I_1^3 - I_2^3}\right) I_2^3 = \frac{C}{D}\left(1 + \frac{4}{\sqrt{I_1}}\right)\left(1 + \frac{6}{\sqrt{I_2}}\right)\left(\frac{1}{1 - (I_2/I_1)^3}\right) I_2^3 .$$

Thus, such a set of delays must exist.

To apply the Lovász Local Lemma, we associate a bad event with each machine. The bad event for machine M is that there is some I_2^3-interval I with a congestion of more than C_2. Let m be the number of operations on M. For every $i \in \{1, \ldots, m\}$, let the random variable X_i denote the length of operation i if operation i is started during I and 0 otherwise. Let $X = \sum_{i=1}^m X_i$. Since the jobs choose their delays uniformly at random from a range of size Δ_2, we get $\Pr[X_i \neq 0] \leq I_2^3/\Delta_2$ for all $i \in \{1, \ldots, m\}$. Hence,

$$\mathrm{E}[X] = \sum_{i=1}^m \mathrm{E}[X_i] \leq \frac{C_1}{I_1^3 - I_2^3} \cdot I_2^3 .$$

Since similar to the first refinement step the X_i's can be grouped together to independent random variables of value at most Q, we get together with the Hoeffding bounds and $\epsilon = \frac{6}{\sqrt{I_2}}$ (note that $\epsilon \leq 1$) that

$$\begin{aligned}
\Pr[X \geq C_2] &= \Pr\left[X \geq (1+\epsilon)\frac{C_1}{I_1^3 - I_2^3} \cdot I_2^3\right] \\
&\leq \mathrm{e}^{-\left(\frac{5}{\sqrt{I_2}}\right)^2 \frac{C_1}{I_1^3 - I_2^3} \cdot I_2^3/3Q} \leq \mathrm{e}^{-12\frac{C}{D}I_2} \leq 2^{-17\frac{C}{D}I_2} .
\end{aligned}$$

Since the total number of I_1^3-intervals to be considered is at most $|F|+\Delta_2 \leq 2I_1^4$, the probability that a bad event occurs for machine M is bounded by

$$p \leq 2I_1^4 \cdot 2^{-17\frac{C}{D}I_2} < \tfrac{D}{C}I_1^{-12} \ .$$

Next we bound the dependence among these events. Clearly, two bad events are independent unless some job has operations at both of the corresponding machines. Since each machine has at most I_1C_1 operations and each of the corresponding jobs has operations in at most I_1^4 other machines, the dependence b of the bad events is at most $C_1I_1^5 \leq 2\frac{C}{D}I_1^8$. Hence, the product $\mathrm{e}p(b+1)$ is less than 1 and therefore, by the Lovász Local Lemma, there is some assignment of delays such that the congestion at every machine in every I_2^3-interval is bounded by C_2.

We continue to refine each frame recursively in a way similar to above. That is, for frames of length I_j^4 we choose delays up to a value of $I_j^3 - I_{j+1}^3$ and bound the congestion in intervals of lengths I_{j+1}^3. Eventually, we reach a value k for which either $I_k = Q$ or $\log I_k < 36$ (implying that I_k is a constant). We end up with a schedule S_k with a total length of at most

$$2D\prod_{j=1}^{k-1}\left(1+\frac{2}{I_j}\right) = O(D)$$

and with a congestion C_k in each I_k^3-frame of at most

$$\frac{C}{D}\cdot I_k^3\left(1+\frac{4}{\sqrt{I_1}}\right)\prod_{j=2}^{k}\left[\left(1+\frac{6}{\sqrt{I_j}}\right)\left(\frac{1}{1-(I_j/I_{j-1})^3}\right)\right] = O\left(\frac{C}{D}\cdot I_k^3\right) \ ,$$

which proves Proposition 7.7. □

This completes the proof of Lemma 7.6. □

7.2.4 An Intermediate Schedule for Acyclic JSS

In this section we prove the existence of an intermediate schedule that fulfills the requirements of Lemma 7.5 with

$$c = \log\log D, \quad \alpha = O(\log\log P), \quad \gamma = O(1 + D/C), \quad \text{and} \quad \beta, \delta = O(1) \ .$$

As in Section 7.2.3 we assume w.l.o.g. that $C \geq D$. Also, we assume that D and $\log D$ are powers of two. It is not difficult to see that if this is not true, rounding these to the next higher power of two does not affect the bounds in this section. To simplify subsequent notation, we reverse the order of the T_i. That is, for every $i \in \{1, \ldots, \frac{\log D}{\log\log D}\}$ the set T_i is defined as $\{\frac{D}{\log^i D}, \ldots, \frac{D}{2\log^{i-1} D}\}$ and $T_0 = \{D\}$. Since there can be at most C/D many T_0-operations on each machine, we will not consider these operations in the following.

Lemma 7.8 *For any acyclic job shop scheduling problem, there exists a well-structured schedule of length $O(D)$ with the following properties holding for every machine:*

- *For every set T_i, the T_i-contention is at most $O(\log\log P + C/D)$.*
- *For every frame F, where $|F|$ is a power of $\log D$, the $(< |F|/\log D)$-congestion is at most $O(|F|C/D)$.*

Proof. We will only prove the existence of a non-well-structured schedule with the properties above. As noted in Lemma 7.6, it can be easily transformed into a well-structured schedule with the same properties.

In case that $P \geq D^{1/4}$, we have $\log\log P = \Theta(\log\log D)$ and therefore can replace the contention bound above by $O(\log\log D + C/D)$. If $P < D^{1/4}$, then we use the same strategy as described for $Q < D^{1/4}$ in Lemma 7.6 to create a schedule that can be partitioned into frames of size P^3 with congestion at most $O(\frac{C}{D} \cdot P^3)$, and continue to schedule each of these frames independently. It therefore suffices to consider the case $P \geq D^{1/4}$, i.e., to show that there exists an intermediate schedule with a T_i-contention of $O(\log\log D + C/D)$ for every i.

Our strategy for constructing an efficient schedule is to make a succession of refinements to an initial schedule S_0. In S_0 each job is executed at every time step until it is completed. This initial schedule is as short as possible; its length is only D. Unfortunately, as many as C jobs may have to use a machine at a single time step in S_0. In order to refine S_0 to a schedule S_1, each job is assigned a suitable delay from the range $[0, D - \frac{D}{\log D} - 1]$. Every job that is assigned a delay of δ waits for δ time steps and then is processed by the machines in the prescribed order until it is completed. Hence, the length of S_1 is bounded by $2D - \frac{D}{\log D}$.

In order to refine S_1 to S_2, each job in S_1 is again assigned a suitable delay, but this time from the range $[0, \frac{D}{\log D} - \frac{D}{\log^2 D} - 1]$. Each job that is assigned a delay of δ waits for δ time steps in addition to the time steps it is already waiting according to S_1 and then is processed by the machines in the prescribed order until it is completed. Hence, the length of S_2 is bounded by $2D$.

For the refinement from S_2 to S_3, we break the schedule of S_2 into frames of length $\frac{D}{\log D}$. Each such frame F_j is considered separately from the other frames, and a subschedule $S_{3,j}$ is constructed for it. Then the subschedules $S_{3,j}$ are concatenated back together (with some overlap) to give schedule S_3.

Let us describe in a bit more detail how to refine a frame F_j to a subschedule $S_{3,j}$. Each operation is associated with the frame at which it starts. Within each frame, each job is given a suitable delay in the range $[0, \frac{D}{\log^2 D} - \frac{D}{\log^3 D} - 1]$. So a job starting at time t in F with delay δ now starts at time $t + \delta$ in $S_{3,j}$. There is no simple bound on the length of $S_{3,j}$, due to long operations that start at F_j and can extend well into subsequent frames. We consider the first $\frac{D}{\log D} + \frac{2D}{\log^2 D}$ steps of a subschedule to be its main part, and the rest of the steps as its extension. Note that all $(< \frac{D}{\log^2 D})$-operations are guaranteed to be completely processed during the main part, whatever their delay is. Schedule S_3 then consists of processing the main parts of $S_{3,1}, S_{3,2}, \ldots$ one after the other, letting the extension of a subschedule run in parallel to the main parts of subschedules that follow it.

The rest of the refinements is similar to the refinement from S_2 to S_3, with the difference that, for a refinement from schedule S_{r-1} to schedule S_r for some $r \geq 4$, S_{r-1} is broken into frames of size $\frac{D}{\log^{r-2} D}$ and within each frame, each job is given a suitable delay in the range $[0, \frac{D}{\log^{r-1} D} - \frac{D}{\log^r D} - 1]$. The main part of any resulting subschedule is considered to be its first $\frac{D}{\log^{r-2} D} + \frac{2D}{\log^{r-1} D}$ steps.

We continue to do these refinements until for the first time, frames of size at most $\log^3 D$ have been refined. Hence, the number of refinement steps is roughly $\frac{\log D}{\log\log D}$. Our refinements lead to the following claim.

Claim 7.9 *For any $r \geq 2$, the total length of schedule S_r is bounded by*

$$\left(1 + \frac{2}{\log D}\right)^{r-2} 2D .$$

This claim shows that the length of the final schedule is at most $3D$ (i.e., $\delta \leq 3$). It remains to prove bounds on the congestion and contention of the operations. We want to achieve this by considering for each $S_{r,j}$ with $r \geq 1$ (let $S_{1,1}$ denote S_1 and $S_{2,1}$ denote S_2) the following problems for every machine:

- For every $i \leq r+1$, we want to bound the contention caused by T_i-operations, and
- for all $(< \frac{D}{\log^{r+1} D})$-operations we want to bound the congestion in any $\frac{D}{\log^r D}$-interval.

For this, let us introduce the following notation. For any time interval I and schedule S, let $S|_I$ denote S restricted to I (that is, $S|_I$ only contains operations that start in I). Fix some machine M and schedule S. Then the *T_i-congestion* C_i^S at M is defined as the congestion at M caused by T_i-operations in S. $C_{\geq i}^S$ denotes the congestion in S caused by all T_j-operations with $j \geq i$. Furthermore, the *T_i-contention* c_i^S at M is defined as the contention at M caused by T_i-operations in S.

In order to obtain the properties above, we prove the following claim.

Claim 7.10 *After the last refinement we end up with a schedule S such that for every machine it holds:*

(1) For every $i \geq 1$, $c_i^S = O(\frac{C}{D} + \log\log D)$, and

(2) for every $r \geq 2$, $C_{\geq r}^{S|_F} = O(\frac{C+D}{\log^{r-2} D})$ for every $\frac{D}{\log^{r-2} D}$-frame F in S.

This implies that $\alpha = O(\log\log D)$ and $\beta, \gamma = O(1)$. Therefore, it remains to show Claim 7.10.

Our refinement strategy above was chosen such that for every refinement step r we can basically use the same analysis for bounding the congestion and contention. Hence, let us consider in the following some fixed $r \geq 1$, and let us assume that for all $r' < r$ the claims below have already been shown to be true. Since our refinement strategy allows us to refine each frame of size $\frac{D}{\log^{r-2} D}$ in S_{r-1} independently, let us consider for the rest of the proof some fixed frame F of this size. Our goal is to refine schedule $S'_{r-1} = S_{r-1}|_F$ to some new schedule S'_r that is then used together with the refined schedules of other frames as described above to construct some schedule S_r.

Bounding the congestion

Let F, S'_{r-1} and S'_r be defined as above. Let $\mathcal{I}_j$ denote the set of all possible time intervals of length $\frac{D}{\log^j D}$ in F, and let $\ell_i = \frac{D}{\log^{i-1} D}$ denote twice the largest possible length of a T_i-operation. The following claim bounds the congestion of S'_r in terms of the congestion of S'_{r-1}.

Claim 7.11 *Let $\hat{C}_i^{r-1} = \max_{I \in \mathcal{I}_{r-1}} C_i^{S'_{r-1}|_I}$, and let $\hat{C}_i^r = \max_{I \in \mathcal{I}_r} C_i^{S'_r|_I}$. There is a schedule S'_r such that (in addition to the contention bounds stated in Claim 7.13) it holds for every $i \geq r+2$ and every machine:*

(1) If $\hat{C}_i^{r-1} \geq \ell_i \log^4 D$ then $\hat{C}_i^r = (1 + O(\frac{1}{\log D}))\frac{\hat{C}_i^{r-1}}{\log D}$.

(2) If $\ell_i \log^3 D \leq \hat{C}_i^{r-1} < \ell_i \log^4 D$ then $\hat{C}_i^r = (1 + O(\frac{1}{\sqrt{\log D}}))\frac{\hat{C}_i^{r-1}}{\log D}$.

(3) If $\ell_i \log^2 D \leq \hat{C}_i^{r-1} < \ell_i \log^3 D$ then $\hat{C}_i^r = O(\frac{\hat{C}_i^{r-1}}{\log D})$.

(4) *If* $\ell_i \log D \leq \hat{C}_i^{r-1} < \ell_i \log^2 D$ *then* $\hat{C}_i^r = O(\ell_i \log D)$.

(5) *If* $\hat{C}_i^{r-1} < \ell_i \log D$ *then* $\hat{C}_i^r \leq \hat{C}_i^{r-1}$.

Furthermore, it holds:

(6) $\max_{I \in \mathcal{I}_{r-1}} C_{r+1}^{S'_r|_I} \leq 2 \max_{I \in \mathcal{I}_{r-1}} C_{r+1}^{S'_{r-1}|_I}$ *and*

(7) $C_r^{S'_r} = C_r^{S'_{r-1}}$.

Obviously, Claim 7.11 (7) is always true, as in both cases we sum up the lengths of exactly the same set of operations. Claim 7.11 (5) and (6) are also true, since the delays for S'_r are chosen from the range $[0, \frac{D}{\log^{r-1} D} - \frac{D}{\log^r D} - 1]$. So it remains to prove Claim 7.11 (1)–(4). This will be done later.

Assume that Claim 7.11 (1)–(4) are true. Then we can show the following lemma.

Lemma 7.12 *For every* $\frac{D}{\log^r D}$*-interval* I *in* S'_r *and every machine,* $C_{\geq r+2}^{S'_r|_I} = O(\frac{C+D}{\log^r D})$.

Proof. Fix some machine M. We start by showing that for every $r \geq 1$ and $i \geq r+2$ it holds for every $\frac{D}{\log^r D}$-interval I in S'_r that

$$C_i^{S'_r|_I} = O\left(\frac{C_i^{S_0}}{\log^r D} + \ell_i \log D\right) .$$

Let us assume that $C_i^{S_0} = \ell_i \log^\alpha D$ for some $\alpha \geq 0$. Then we get for every $1 \leq r \leq \alpha - 3$ and every $\frac{D}{\log^r D}$-interval I in S'_r with the help of Claim 7.11 (1) that

$$C_i^{S'_r|_I} = \left(1 + O\left(\frac{1}{\log D}\right)\right)^r \frac{C_i^{S_0}}{\log^r D} .$$

For $\alpha - 3 < r \leq \alpha - 2$ and every $\frac{D}{\log^r D}$-interval I in S'_r, this yields with Claim 7.11 (2)

$$C_i^{S'_r|_I} = \left(1 + O\left(\frac{1}{\log D}\right)\right)^{r-1} \left(1 + O\left(\frac{1}{\sqrt{\log D}}\right)\right) \frac{C_i^{S_0}}{\log^r D} ,$$

and for $\alpha - 2 < r \leq \alpha - 1$ and every $\frac{D}{\log^r D}$-interval I in S'_r, this yields with Claim 7.11 (3)

$$\begin{aligned} C_i^{S'_r|_I} &= \left(1 + O\left(\frac{1}{\log D}\right)\right)^{r-2} \left(1 + O\left(\frac{1}{\sqrt{\log D}}\right)\right) O\left(\frac{C_i^{S_0}}{\log^r D}\right) \\ &= O\left(\frac{C_i^{S_0}}{\log^r D}\right) . \end{aligned}$$

Finally, for $\alpha - 1 < r \leq \alpha$ we get with Claim 7.11 (4) that $C_i^{S'_r|_I} = O\left(\ell_i \log D\right)$.

So for every $r \geq 1$ and every $\frac{D}{\log^r D}$-interval I in S'_r, $C_{\geq r+2}^{S'_r|_I}$ is bounded by

$$\begin{aligned} \sum_{i \geq r+2} C_i^{S'_r|_I} &= \sum_{i \geq r+2} O\left(\frac{C_i^{S_0}}{\log^r D} + \ell_i \log D\right) \\ &= \frac{1}{\log^r D} \sum_{i \geq r+2} O(C_i^{S_0}) + \log D \sum_{i \geq r+2} O\left(\frac{D}{\log^{i-1} D}\right) \\ &= O\left(\frac{C+D}{\log^r D}\right) . \end{aligned}$$

□

Hence, for every machine M, the total congestion in S'_r caused by $(< \frac{D}{\log^{r-1} D})$-operations in frame F is bounded by

$$\underbrace{(\log D)^2 \cdot O\left(\frac{C+D}{\log^r D}\right)}_{(< \frac{D}{\log^{r+1} D})\text{-operations}} + \underbrace{(\log D) \cdot O\left(\frac{C+D}{\log^{r-1} D}\right)}_{T_{r+1}\text{-operations}} + \underbrace{O\left(\frac{C+D}{\log^{r-2} D}\right)}_{T_r\text{-operations}} = O\left(\frac{C+D}{\log^{r-2} D}\right) .$$

The first term in the left hand side comes from Lemma 7.12 together with the fact that there are roughly $(\log D)^2$ disjoint intervals of length $D/\log^r D$ in frame F. The second term uses Lemma 7.12 and Claim 7.11 (6), and the third term uses Lemma 7.12, Claim 7.11 (6), and Claim 7.11 (7). Hence, under the assumption that Claim 7.11 is correct, Claim 7.10 (2) follows. It remains to prove Claim 7.10 (1).

Bounding the contention

In order to bound the contention, we will prove the following claim.

Claim 7.13 *There is a schedule S'_r such that (in addition to the congestion bounds stated in Claim 7.11)*

(1) $c_{r+1}^{S'_r} = O(\frac{C}{D} + \log\log D)$,

(2) $c_r^{S'_r} = O(c_r^{S'_{r-1}})$, *and*

(3) $c_i^{S'_r} \leq 2c_i^{S_i}$ *for all* $i \leq r-1$.

Claim 7.13 (3) is easy to show. Since the total amount of time steps by which the T_i-operations move for the refinements following schedule S_i is bounded by

$$\sum_{j>i} \left(\frac{D}{\log^{j-1} D} - \frac{D}{\log^j D}\right) < \frac{D}{\log^i D} = \frac{\ell_i}{\log D} ,$$

the contention of these operations can at most double afterwards. Suppose that Claim 7.13 (1) and (2) are also true. Then we can show the following lemma.

Lemma 7.14 *After the last refinement, for every $i \geq 1$ and machine, the contention among the T_i-operations is bounded by $O(\frac{C}{D} + \log\log D)$.*

Proof. Using Claim 7.13 it follows that, for any $r \geq 1$ and $i \leq r+1$, the contention among the T_i-operations is bounded by $O(\frac{C}{D} + \log\log D)$ in S_r. We mentioned earlier that we want to stop the refinements after for the first time frames of size at most $\log^3 D$ have been refined, that is, $\frac{D}{\log^{r-2} D} \leq \log^3 D$. In this case, T_{r+1} represents a set of t-operations with $t \in \{1, \ldots, \frac{1}{2}\log D\}$, and since their contention is bounded because of Claim 7.13 (1), the lemma follows. □

So if Claim 7.13 is true, then Claim 7.10 (1) follows.

Proofs of Claim 2.10 and Claim 2.12

Recall that we want to show that on frame F of length $D/\log^{r-2} D$ we can refine schedule S'_{r-1} to obtain a schedule S'_r that satisfies certain bounds on the congestion and contention. Our proof will use the general form of the Lovász Local Lemma. We associate bad events with each machine. For some fixed machine M, A_i denotes the event that the congestion of T_i-operations is too high in S'_r, and B_i denotes the event that the contention of T_i-operations is too high. We first bound the probabilities for the A_i and B_i events.

Let us consider some event A_i with $i \geq r+2$, denoted by A in the following. We distinguish between the following two cases:

- $\hat{C}_i^{r-1} \geq \ell_i \log^2 D$: Consider some fixed $\frac{D}{\log^{r-1} D}$-interval I in S'_{r-1}. The interval composed of the last $\frac{D}{\log^r D}$ time steps of I is called the *tail* of I. Let all T_i-operations on M that start in I be numbered from 1 to a. For every $j \in \{1, \ldots, a\}$, let the random variable X_j be t if operation j is a t-operation and chooses a delay such that it starts during the tail of I, and 0 if operation j chooses a delay such that it starts either before or after the tail of I. (Note that all operations that might be processed during the tail of I in S'_r start in S'_{r-1} during I.) Let $X = \sum_{j=1}^{a} X_j$. Since the jobs choose their delays independently and uniformly at random from a range of $[0, \frac{D}{\log^{r-1} D} - \frac{D}{\log^r D} - 1]$, we get

$$\Pr[X_j \neq 0] \leq \frac{D/\log^r D}{D/(\log^{r-1} D) - D/(\log^r D)} = \frac{1}{\log D - 1}$$

for every j. Hence, $\mathrm{E}[X] \leq \hat{C}_i^{r-1}/(\log D - 1)$. Let $\mu = \hat{C}_i^{r-1}/(\log D - 1)$ and $\delta = \min[\sqrt{\hat{C}_i^{r-1}/(\ell_i \log^2 D)},\ \log D]$. Since the operations considered for A have a length of at most $k = \ell_i/2$, we get together with the Hoeffding bounds that, for $\epsilon = \frac{3}{\delta}$ if $\frac{3}{\delta} \leq 1$ and $\epsilon = (\frac{3}{\delta})^2$ otherwise,

$$\Pr\left[X \geq (1+\epsilon)\mu\right] \leq \mathrm{e}^{-(3/\delta)^2 \frac{\hat{C}_i^{r-1}}{\log D - 1}/3k} \leq \mathrm{e}^{-6 \max\left[\log D,\ \frac{\hat{C}_i^{r-1}}{\ell_i \log^3 D}\right]} .$$

In this bound, we replace the expression $\hat{C}_i^{r-1}/(\ell_i \log^3 D)$ as follows. Let m_A denote the number of T_i-operations on M in the frame F. Since the smallest T_i-operations have size $\ell_i/\log D$ and frame F has $\log D$ disjoint intervals of size $D/(\log D)^{r-1}$, we have that $m_A \leq (\log D \cdot \hat{C}_i^{r-1})/(\ell_i/\log D)$ and therefore $\hat{C}_i^{r-1} \geq m_A \cdot \ell_i/\log^2 D$. We conclude that for ϵ as defined in the two cases above

$$\Pr\left[X \geq (1+\epsilon)\frac{\hat{C}_i^{r-1}}{\log D - 1}\right] \leq \mathrm{e}^{-6\max\left[\log D,\ m_A/\log^5 D\right]} .$$

Let us define the event A to be true if $X \geq (1+\epsilon)\frac{\hat{C}_i^{r-1}}{\log D - 1}$ for some $\frac{D}{\log^{r-1} D}$-interval in S'_{r-1}. Since at most $2\frac{D}{\log^{r-2} D}$ different $\frac{D}{\log^{r-1} D}$-intervals have to be considered, we get

$$\Pr[A] \leq \min\left[\frac{1}{D^5},\ D \cdot \mathrm{e}^{-6 m_A/\log^5 D}\right] .$$

- $\ell_i \log D \leq C_i^{r-1}(I) < \ell_i \log^2 D$: Let X, μ, and k be defined as before. Then we get with $\epsilon = \frac{6 \log D}{\mu/k}$ that

$$\Pr[X \geq (1+\epsilon)\mu] \leq \left(\frac{\mathrm{e}^{\epsilon}}{(1+\epsilon)^{1+\epsilon}}\right)^{\mu/k} \leq \mathrm{e}^{-6 \log D} \leq D^{-6} .$$

The rest is similar to above to obtain $\Pr[A] \leq \frac{1}{D^5}$.

Next we bound the probabilities for the contention events. For some fixed machine M and $i \in \{r, r+1\}$, let us consider the respective event B_i, which we will denote by B in the following.

First, we consider the case that $i = r+1$. Consider some fixed $\frac{D}{\log^{r-1} D}$-interval I in S'_{r-1}, and let t denote its last time step. Let O_I denote the set of all T_{r+1}-operations on M that start in I. (Note that O_I contains all T_{r+1}-operations that might be processed at time step t in schedule S'_r.) Since each unit of processing of an operation in O_I has a probability of at most $1/(\frac{D}{\log^{r-1} D} - \frac{D}{\log^r D})$ to be executed at time t, the probability that $k = \mathrm{e}(\mathrm{e}+1)\max[\hat{C}^{r-1}_{r+1}/|I|,\ \log\log D]$ operations are executed at time t is bounded by

$$\binom{\hat{C}^{r-1}_{r+1}}{k}\left(\frac{1}{\frac{D}{\log^{r-1} D} - \frac{D}{\log^r D}}\right)^k \le \left(\frac{\mathrm{e}\hat{C}^{r-1}_{r+1}}{k\left(1 - \frac{1}{\log D}\right)|I|}\right)^k \le \mathrm{e}^{-k} .$$

Let us define B to be true if this contention bound is true for the last time step of every $\frac{D}{\log^{r-1} D}$-interval I in S'_{r-1}. Using Lemma 7.12 as an inductive hypothesis for refinement step $r-1$, $\hat{C}^{r-1}_{r+1} = O(\frac{C+D}{\log^{r-1} D})$. Thus,

$$k = O\left(\frac{C+D}{\log^{r-1} D} \cdot \frac{1}{|I|} + \log\log D\right) = O(C/D + \log\log D) ,$$

as desired. Since the smallest operations in B have a length of at least $\ell_{r+1}/\log D$, the total number m_B of T_{r+1}-operations on M within frame F is at most $(\log D \cdot \hat{C}^{r-1}_{r+1})/(\ell_{r+1}/\log D)$ and therefore,

$$\hat{C}^{r-1}_{r+1} \ge \frac{m_B \cdot \ell_{r+1}}{\log^2 D} = m_B \cdot \frac{D}{\log^{r+2} D} .$$

Hence, the probability that B is true w.r.t. I is at most $\mathrm{e}^{-10\max[\log\log D,\ m_B/\log^3 D]}$. Since at most $2\log^3 D$ time steps have to be checked to be sure that nowhere in S'_{r-1} the contention caused by T_{r+1}-operations exceeds k, we get

$$\Pr[B] \le \min\left[\frac{1}{\log^{10} D},\ 2\log^3 D \cdot \mathrm{e}^{-10 m_B/\log^3 D}\right] .$$

The calculations for the case that event B considers T_r-operations are similar. If we assume Claim 7.13 (1) to be true for these operations after refinement step $r-1$ and choose $k = O(\max[\hat{C}^{r-1}_r/\frac{D}{\log^{r-1} D},\ \log\log D])$ large enough, then we end up with the same probability and contention bounds as above.

Next we bound the dependencies among the A_i-events and the B_j-events. Whether or not a bad event A_i occurs depends solely on the delays assigned to the jobs that have a T_i-operation on the respective machine M. Each of these m_{A_i} jobs may have operations that influence up to D other A_j on other machines. Hence, each A_i depends on at most $m_{A_i} D$ other A_j-events. Furthermore, every such job may also have operations that influence at most $\log^3 D$ B_j-events on other machines. This holds, since schedule S'_{r-1} is based on a frame of size $\frac{D}{\log^{r-2} D}$ in S_{r-1} and the lengths of the operations influencing B_j-events are at least $\frac{D}{\log^{r+1} D}$. Hence, the maximum dependence of an A_i-event on B_j-events is bounded by $m_{A_i}\log^3 D$.

Whether or not a bad event B_i occurs also depends solely on the delays assigned to the jobs that have a T_i-operation on the respective machine M. Since each of these m_{B_i} jobs may have operations that influence up to $\log^3 D$ other B_j, each B_i depends on at most $m_{B_i}\log^3 D$ other

B_j-events. Furthermore, every such job may also influence up to D A_j-events. Hence, the maximum dependence of a B_i-event on A_j-events is bounded by $m_{B_i}D$.

Before we use the Lovász Local Lemma, we state a simple claim

Claim 7.15 *For any* $0 \leq x \leq 1/2$, $1 - x \geq \mathrm{e}^{-2x}$.

Proof. Since for all $0 \leq x \leq 1/2$ we have $1 - 2x + 2x^2 \geq \mathrm{e}^{-2x}$ and $1 - x \geq 1 - 2x + 2x^2$, the claim follows. □

Now we are ready to apply the Lovász Local Lemma. We will assume that D is sufficiently large. For every A_i, choose p_{A_i} as $p_{A_i} = \frac{1}{D^3}$, and for every B_i, choose p_{B_i} as $p_{B_i} = \frac{1}{\log^9 D}$. Let G be the dependency graph of the A_i-events and B_j-events. Then it holds for every A_i that

$$\begin{aligned} p_{A_i} \prod_{\{A_i,A_j\}\in G} (1 - p_{A_j}) \prod_{\{A_i,B_j\}\in G} (1 - p_{B_j}) &\geq \frac{1}{D^3}\left(1 - \frac{1}{D^3}\right)^{m_{A_i}D}\left(1 - \frac{1}{\log^9 D}\right)^{m_{A_i}\log^3 D} \\ &\geq \frac{1}{D^3} \cdot \mathrm{e}^{-2m_{A_i}/D^2} \cdot \mathrm{e}^{-2m_{A_i}/\log^6 D} \geq \Pr[A_i] , \end{aligned}$$

because for $m_{A_i} \leq \log^6 D$ we have

$$\frac{1}{D^3} \cdot \mathrm{e}^{-2m_{A_i}/D^2} \cdot \mathrm{e}^{-2m_{A_i}/\log^6 D} \geq \frac{1}{D^5}$$

and for $m_{A_i} > \log^6 D$ we have

$$\frac{1}{D^3} \cdot \mathrm{e}^{-2m_{A_i}/D^2} \cdot \mathrm{e}^{-2m_{A_i}/\log^6 D} \geq D \cdot \mathrm{e}^{-6m_{A_i}/\log^5 D} .$$

For every B_i we further get that

$$\begin{aligned} p_{B_i} \prod_{\{B_i,A_j\}\in G} (1 - p_{A_j}) \prod_{\{B_i,B_j\}\in G} (1 - p_{B_j}) &\geq \frac{1}{\log^9 D}\left(1 - \frac{1}{D^3}\right)^{m_{B_i}D}\left(1 - \frac{1}{\log^9 D}\right)^{m_{B_i}\log^3 D} \\ &\geq \frac{1}{\log^9 D} \cdot \mathrm{e}^{-2m_{B_i}/D^2} \cdot \mathrm{e}^{-2m_{B_i}/\log^6 D} \geq \Pr[B_i] , \end{aligned}$$

because for $m_{B_i} \leq \log^4 D$ we have

$$\frac{1}{\log^9 D} \cdot \mathrm{e}^{-2m_{B_i}/D^2} \cdot \mathrm{e}^{-2m_{B_i}/\log^6 D} \geq \frac{1}{\log^{10} D}$$

and for $m_{B_i} > \log^4 D$ we have

$$\frac{1}{\log^9 D} \cdot \mathrm{e}^{-2m_{B_i}/D^2} \cdot \mathrm{e}^{-2m_{B_i}/\log^6 D} \geq 2\log^3 D \cdot \mathrm{e}^{-10m_{B_i}/\log^3 D} .$$

Hence, according to the Lovász Local Lemma, there exists a set of delays that fulfills the requirements of Claims 7.11 and 7.13. So a refinement from S_{r-1} to S_r is possible as we claimed it, which proves Lemma 7.8. □

7.2.5 Proof of Item 2 of Theorem 1.1

As before, let us assume that $C \geq D$. In order to simplify the proof, we first consider the situation that there are only two different operation lengths, P_1 and P_2, with $P_1 > P_2$ (recall that w.l.o.g. we can assume that P_1 and P_2 are powers of 2). Our strategy for constructing an efficient schedule is to successively refine the (possibly illegal) well-structured schedule S_0 that runs all jobs in parallel. Let $I_0 \cdot P_1$ with $I_0 = D/P_1$ be the frame size to be refined in schedule S_0 and $I_j^3 P_1$ with $I_j = \log I_{j-1}$ for all $j \geq 1$ be the frame size to be refined in schedule S_j. Our aim is to show that for all $j \geq 1$ with $I_j \geq 36$ there exists a refinement from schedule S_{j-1} to a well-structured schedule S_j such that the congestion at any machine for any frame of size $I_j^2 P_1$ is bounded by $O(\frac{C}{D} \cdot I_j^2 P_1)$. (The condition $I_j \geq 36$ was only chosen to simplify the proof.) So when $\log I_j < 36$ for the first time then the contention among the P_1-operations must be $O(C/D)$. Furthermore, the congestion caused by P_2-operations in any P_1-interval must be bounded by $O(\frac{C}{D} \cdot P_1)$. Our strategy therefore is to break schedule S_j into sub-schedules of length P_1 and continue to schedule each sub-schedule independently, using the same refinement strategy for the P_2-operations as before for the P_1-operations. At the end we want to arrive at a well-structured schedule of length $O(D)$ in which both P_1-operations and P_2-operations have a contention of $O(C/D)$. This schedule can then be transformed into a legal schedule of length $O(C + D)$ by using the strategy in the proof of Lemma 7.5 (1).

Let us describe now in detail, how to do the refinements. In the following, set $d_1 = D/P_1$ and $d_2 = D/P_2$. The first step is to assign an initial delay to each job, chosen independently and uniformly at random from $\{iP_1 \mid i \in [0, d_1 - 1]\}$. In the resulting schedule, S_1, a job that is assigned a delay of δ waits for δ steps and then is processed without waiting again until it is completed. The length of S_1 is at most $2D$. We show that if the delays are chosen independently and uniformly at random and I_1 is sufficiently large, then with nonzero probability the congestion at any machine in any time interval of size $I_1^2 P_1$ starting at an integer multiple of P_1 is at most $C_1 = \frac{C}{D}(1 + \frac{4}{\sqrt{I_1}}) I_1^2 P_1$ for both P_1- and P_2-operations. (Since the delays are multiples of P_1 and S_0 is well-structured, we only have to consider these $I_1^2 P_1$-intervals for our refinement step.) Thus, such a set of delays must exist.

To apply the Lovász Local Lemma, we associate two bad events with each machine. Fix some machine M. For $i \in \{1, 2\}$, the bad event $A_{M,i}$ for the P_i-operations at M is that there is some $I_1^2 P_1$-interval starting at an integer multiple of P_1 with a congestion of more than C_1. For this we have to bound the dependence among the bad events and the probability that a bad event occurs.

Let us start with bounding the probability. Consider some fixed event $A_{M,i}$, denoted by A, and $I_1^2 P_1$-interval I. Let m_i be the number of P_i-operations on M. For every $j \in \{1, \ldots, m_i\}$, let the random variable X_j be P_i if operation j is started in I and 0 otherwise. Let $X = \sum_{j=1}^{m_i} X_j$. Since the jobs choose their delays uniformly at random from $\{iP_1 \mid i \in [0, d_1 - 1]\}$, we get $\Pr[X_j = P_i] \leq I_1^2 / d_1$ for all j. Hence,

$$\mathrm{E}[X] = \sum_{j=1}^{m_i} \mathrm{E}[X_j] \leq \frac{C}{D} \cdot I_1^2 P_1 \ .$$

Let $c_1 = C/P_1$ and $c_2 = C/P_2$. Together with the Hoeffding bounds we get with $\epsilon = \frac{4}{\sqrt{I_1}}$ (note that $I_1 \geq 36$) that

$$\begin{aligned} \Pr[X \geq C_1] &= \Pr\left[X \geq (1+\epsilon)\tfrac{C}{D} \cdot I_1^2 P_1\right] \\ &\leq \mathrm{e}^{-\left(\frac{4}{\sqrt{I_1}}\right)^2 \frac{C}{D} I_1^2 P_1 / 3P_i} = \mathrm{e}^{-5 I_1 \frac{C/P_i}{D/P_1}} \end{aligned}$$

$$= \mathrm{e}^{-5I_1 c_i/d_1} .$$

The total number of $I_1^2 P_1$-intervals to be considered is at most $2d_1$. Hence, the probability that a bad event A occurs is bounded by

$$2d_1 \cdot \mathrm{e}^{-5I_1 c_i/d_1} = 2d_1 \cdot \mathrm{e}^{-5(\log d_1)c_i/d_1} \le d_1^{-5c_i/d_1} .$$

Now, we bound the dependencies among the events. Whether or not a bad event $A_{M,1}$ occurs depends solely on the delays assigned to the jobs whose operations influence $A_{M,1}$. Since each of the at most c_1 jobs regarded by $A_{M,1}$ may have operations that influence up to d_1 other $A_{M',1}$, each $A_{M,1}$ depends on at most $c_1 d_1$ other $A_{M',1}$-events. Furthermore, every job associated with $A_{M,1}$ may also have operations that influence up to d_2 $A_{M',2}$-events. Hence the maximum dependence of an $A_{M,1}$-event on $A_{M',2}$-events is bounded by $c_1 d_2$. Similarly, each $A_{M,2}$ depends on at most $c_2 d_2$ other $A_{M',2}$-events and on at most $c_2 d_1$ other $A_{M',1}$-events.

Now we are ready to apply the Lovász Local Lemma. Recall that, since $C \ge D$, $c_1 \ge d_1$ and $c_2 \ge d_2$. Let $p_1 = d_1^{-2c_1/d_1}$ and $p_2 = d_1^{-2c_2/d_1}$. Let G be defined as the dependency graph of the $A_{M,1}$-events and $A_{M,2}$-events. Then it holds for every $A_{M,1}$ that

$$\begin{aligned} & p_1 \prod_{\{A_{M,1},A_{M',1}\}\in G} (1-p_1) \prod_{\{A_{M,1},A_{M',2}\}\in G} (1-p_2) \\ \ge\ & d_1^{-2c_1/d_1} \left(1 - d_1^{-2c_1/d_1}\right)^{c_1 d_1} \left(1 - d_1^{-2c_2/d_1}\right)^{c_1 d_2} \\ >\ & d_1^{-2c_1/d_1} \cdot \mathrm{e}^{-2d_1^{-2c_1/d_1} c_1 d_1} \cdot \mathrm{e}^{-2d_1^{-2c_2/d_1} c_1 d_2} \\ \ge\ & d_1^{-2c_1/d_1} \cdot \mathrm{e}^{-2} \cdot \mathrm{e}^{-2c_1/d_1} \ge \Pr\left[A_{M,1}\right] , \end{aligned}$$

because

$$d_1^{-2c_1/d_1} \cdot c_1 d_1 \le 1 \quad \Leftrightarrow \quad 2c_1 \ge d_1 \left(1 + \log_{d_1} c_1\right) ,$$

which is true for all $c_1 \ge d_1$, and

$$d_1^{-2c_2/d_1} \cdot c_1 d_2 \le c_1/d_1 \quad \Leftrightarrow \quad 2c_2 \ge d_1 \left(1 + \log_{d_1} d_2\right) ,$$

which is also true since $c_2 \ge d_2 \ge d_1$. For every $A_{M,2}$ we further get that

$$\begin{aligned} & p_2 \prod_{\{A_{M,2},A_{M',1}\}\in G} (1-p_1) \prod_{\{A_{M,2},A_{M',2}\}\in G} (1-p_2) \\ \ge\ & d_1^{-2c_2/d_1} \left(1 - d_1^{-2c_1/d_1}\right)^{c_2 d_1} \left(1 - d_1^{-2c_2/d_1}\right)^{c_2 d_2} \\ >\ & d_1^{-2c_2/d_1} \cdot \mathrm{e}^{-2d_1^{-2c_1/d_1} c_2 d_1} \cdot \mathrm{e}^{-2d_1^{-2c_2/d_1} c_2 d_2} \\ \ge\ & d_1^{-2c_2/d_1} \cdot \mathrm{e}^{-2c_2/d_1} \cdot \mathrm{e}^{-2} \ge \Pr\left[A_{M,2}\right] , \end{aligned}$$

because

$$d_1^{-2c_1/d_1} \cdot c_2 d_1 \le c_2/d_1 \quad \Leftrightarrow \quad c_1 \ge d_1 ,$$

which is true, and

$$d_1^{-2c_2/d_1} \cdot c_2 d_2 \le 1 \quad \Leftrightarrow \quad 2c_2 \ge d_1 \left(1 + \log_{d_1} d_2\right) ,$$

which is also true since $c_2 \ge d_2 \ge d_1$. Hence, there is an assignment of delays to the jobs such that a refinement from schedule S_0 to S_1 is possible such that the congestion at any machine M_i in any $I_1^2 P_1$-interval is bounded by C_1.

Next, we break schedule S_1 into frames of size $I_1^3 P_1$ and schedule each frame independently. So each frame can be viewed as a separate scheduling problem where the jobs associated with a frame are those that have operations scheduled by S_1 within the frame, and each such job contains only its operations scheduled within that frame. For each frame, our refinement step will choose for each associated job a suitable initial delay. Schedule S_2 then results from executing the modified frames one after the other.

Consider some fixed $I_1^3 P_1$-frame F in S_1. Let each job randomly choose an (additional) initial delay in the range $\{iP_1 \mid i \in [0, I_1^2 - 1]\}$. Then the length of the resulting schedule for this frame is at most $I_1^3 P_1 + I_1^2 P_1 = (1 + \frac{1}{I_1}) I_1^3 P_1$. It is not difficult to see that we can use the Lovász Local Lemma in a similar way as above to show that if the delays are chosen independently and uniformly at random, then with nonzero probability the congestion in any $I_2^2 P_1$-interval in F starting at an integer multiple of P_1 is at most

$$C_2 = \frac{C}{D}\left(1 + \frac{4}{\sqrt{I_1}}\right)\left(1 + \frac{6}{\sqrt{I_2}}\right)\left(\frac{1}{1 - (I_2/I_1)^2}\right) I_2^2 P_1$$

for both P_1- and P_2-operations. Thus, such a set of delays must exist. (The analysis is a combination of the analysis above and the second refinement step in the proof of Proposition 7.7).

Continuing with the refinements until $\log I_j < 36$ completes the refinements for the P_1-operations. The processing of the P_2-operations is afterwards refined similar to Proposition 7.7. This proves the theorem for two different operation lengths.

Extending this refinement technique from two different operation lengths to any constant number of operation lengths by using it again and again from the largest to the smallest operations proves item 2 of Theorem 7.1.

7.3 Lower Bounds

We first exhibit an instance of general JSS for which it is relatively simple to show that $L = \Omega(lb \log lb / \log\log lb)$.

7.3.1 General Job Shop Scheduling

Theorem 7.16 *For any integers $1 < k < r$, there is an instance of job shop scheduling with k jobs, k machines, dilation and congestion each equal to $D = kr^{k-1}$, such that any legal schedule is of length more than $Dk(1 - \frac{k-1}{r-1})$.*

Proof. We describe the JSS instance for arbitrary integers $1 < k < r$. There are k machines. Each job is composed of repetitions of the sequence of operations $M_1, M_2, \ldots, M_k$. For job i, $1 \leq i \leq k$, each operation requires time r^{i-1}, and the sequence as a whole is repeated r^{k-i} times. Hence, the length of each job (the dilation) is $r^{i-1} \cdot k \cdot r^{k-i} = kr^{k-1}$, which we denote by D. The congestion (the total demand for any individual machine) is also D, as each job requires time D/k on each machine.

Fix an arbitrary legal schedule S for the jobs, without preemption. We shall show that the length of the schedule must be more than $Dk(1 - \frac{k-1}{r-1})$.

Consider two arbitrary jobs, J_i and J_j. Relative to the schedule S, we define the overlap $O(i, j)$ as the number of time units in which both J_i and J_j perform operations.

Proposition 7.17 *For* $1 \le i < j \le k$,

$$O(i,j) \le \frac{D(k-1)}{r^{j-i}}$$

Proof. During the execution of an operation of J_j on machine M, job J_i can complete at most $k-1$ operations (as operations on machine M have to wait). Each operation of J_i is shorter than the operation of J_j by a factor of r^{j-i}. □

We now use the first two terms of the exclusion-inclusion formula to bound $|S|$, the total length of schedule S. We obtain, $|S| \ge kD - \sum_{1\le i<j\le k} O(i,j)$. But

$$\sum_{1\le i<j\le k} O(i,j) = \sum_{2\le j\le k}\sum_{1\le i<j} O(i,j) \le \sum_{2\le j\le k}\sum_{1\le i<j} \frac{D(k-1)}{r^{j-i}} < \sum_{1\le j\le k} \frac{D(k-1)}{r-1} = \frac{k(k-1)D}{r-1}$$

Hence, $|S| > kD(1-\frac{k-1}{r-1})$. □

Observe that already for the case of three jobs and three machines, when r is sufficiently large we get that every job and every machine is idle for most of the duration of any legal schedule. Also, when $r \simeq k\log k$, then the length of the shortest schedule is nearly kD, and $k \simeq \log D/\log\log D$. This proves Theorem 7.3.

If preemption is allowed, then the jobs in the proof of Theorem 7.16 can be scheduled within $O(D)$ steps. This is a consequence of item 3 in Lemma 7.5.

7.3.2 Acyclic Job Shop Scheduling

Here we prove Theorem 7.4, which is a direct consequence of Theorem 7.22 below.

Throughout this section we assume that m is sufficiently large. In order to avoid excessive use of floor and ceiling notation, we treat all functions of m (such as $\log m$, $\sqrt{m}$, $5m/6\log m$, etc.) as if they give integer values. (The effect of rounding fractional values of these functions to the nearest integer is negligible for the purpose of our lower bound, when m is sufficiently large.) We start with preliminary definitions and a lemma.

Definition 7.18 *A* bucket configuration $B(m,k)$ *is a collection of* 2^k *buckets, where each bucket contains* $m - m/(\log m)^2$ *of the* m *machines.*

Definition 7.19 *A* partial cover *of job* $J = M_{i_1}, M_{i_2}, \ldots, M_{i_l}$ *by bucket configuration* $B(m,k)$ *is the assignment of the buckets of* $B(m,k)$ *to non-overlapping intervals within* $i_1, \ldots i_l$*, such that within each interval, all operations of job J are on machines that are in the respective bucket. Job J* avoids *bucket configuration* $B(m,k)$ *if for any partial cover of J by* $B(m,k)$*, the number of operations of J that remain uncovered is at least* $l/2$.

Definition 7.20 *A* bucket avoiding family of jobs $F(l,m,k)$ *is a collection of* $m/\log m$ *jobs, where each job contains* 2^l *operations and the collection has the following properties:*

(1) Each machine participates in at most $2^{l+1}/\log m$ *jobs.*

(2) Each job is acyclic.

(3) For any bucket configuration $B(m,k)$*, at least half of the jobs avoid it.*

Lemma 7.21 *For m sufficiently large, for $k \leq \frac{\log m}{2} - 4\log\log m - 1$, and for $l = k + 4\log\log m$, there is a bucket avoiding family of jobs $F(l, m, k)$.*

Proof. We shall use the term *overwhelming* (*negligible*, respectively) to denote a probability that tends to 1 (to 0, respectively) as m tends to infinity.

Consider the following four step randomized procedure for constructing $F(l, m, k)$.

(1) For each of the $m/\log m$ jobs independently, each of the 2^l operations in a job is to be executed on a machine chosen uniformly at random, independently of all other choices.

(2) If there is some machine that participates in more than $2^{l+1}/\log m$ operations, abort the construction.

(3) If less than $5m/6\log m$ jobs are acyclic, abort.

(4) Leave $5m/6\log m$ acyclic jobs untouched. The total number of operations in the remaining jobs is $2^l m/6\log m$. Globally rearrange these operations within the $m/6\log m$ jobs so that each of these jobs becomes acyclic. (By item 2 above, each machine participates in at most $2^{l+1}/\log m$ operations. Observe that for sufficiently large m, the condition that $l < \frac{\log m}{2}$ implies that $2^{l+1}/\log m < m/6\log m$. Hence, the rearranging is possible.)

First, let us show that the probability that the construction aborts is negligible.

View each operation as a ball and each machine as a bin. Then choosing an operation at random is the same as putting a ball in a random bin. We have $m2^l/\log m$ balls and m bins. For $l \geq 4\log\log m$ (as we have in the lemma), the expected number of balls per bin is at least $(\log m)^3$, and standard analysis of throwing balls into bins shows that the probability that some bin contains more balls than twice the expectation is negligible. Hence, the probability of aborting in step 2 of the construction is negligible.

Consider any particular job. The probability that it has two operations on the same machine is at most $\binom{2^l}{2}/m$, which is less than $1/8$, as $l \leq \frac{\log m}{2} - 1$. Hence, the expected number of acyclic jobs is at least $7m/8\log m$, and standard bounds on large deviations show that the probability that step 3 aborts is negligible.

We conclude that with overwhelming probability, the construction does not abort. Observe that in this case, conditions 1 and 2 of definition 7.20 hold, and it remains to verify that condition 3 holds with high probability. We shall check a stronger condition after step 1 of the construction, namely, that for any bucket configuration $B(m, k)$, at least two thirds of the jobs avoid it. This implies that condition 3 holds after step 4 of the construction, as only one sixth of the jobs are rearranged.

Fix a particular bucket configuration $B(m, k)$, and consider a job J composed of a sequence of 2^l machines chosen independently at random. If J does not avoid $B(m, k)$, then 2^{l-1} of its operations are covered by the 2^k buckets. Hence, there are 2^k disjoint sequences of $s = 2^{l-k-2}$ consecutive operations such that each one of them is completely covered by one bucket (though the same bucket can cover several such sequences). There are 2^l ways of choosing the starting point of a sequence, and for each sequence there is a choice of 2^k buckets that may cover it. For a particular starting point and a particular bucket, the probability that all of the s random machines in the sequence are among the $m - m/(\log m)^2$ machines in the bucket is $(1 - (\log m)^{-2})^s$. Hence, we conclude that the probability that a random job fails to avoid a particular bucket configuration is at most $(2^{k+l}(1 - (\log m)^{-2})^s)^{2^k}$. Substituting back in the value of s, setting $l = k + 4\log\log m$, and using the fact that $l \leq \frac{\log m}{2}$, this can be upper bounded by $(2^{2k}2^{4\log\log m}e^{-(\log m)^2+4})^{2^k} < e^{-\alpha(\log m)^2 2^k}$, for some α that can be made arbitrarily close to 1 when m is sufficiently large.

If there are $m/\log m$ random jobs as above, the probability that a third of them fail to avoid the bucket configuration is $2^{-\beta m \log m 2^k}$ for some $\beta > 0$. As there are less than $m^{m2^k/(\log m)^2} = 2^{m2^k/\log m}$ possible bucket configurations, the probability that a third of the jobs fail to avoid some bucket configuration is negligible. This takes care of condition 3 in definition 7.20.

As the randomized construction has overwhelming probability of succeeding, we have shown that a bucket avoiding family $F(l, m, k)$ exists. □

Theorem 7.22 *For a large enough m, there is an instance of acyclic JSS, with m machines, less than m jobs, dilation roughly $\sqrt{m}$, congestion $o(\sqrt{m})$, for which any legal schedule requires at least $\frac{\sqrt{m}\log m}{32\log\log m}$ time steps. Expressed in terms of dilation (in our case, the congestion is much smaller than the dilation), any legal schedule has a length of at least $D\frac{\log D}{16\log\log D}$.*

Proof. Let m be sufficiently large. The instance is composed of $K = (\log m - 2)/4\log\log m + 1$ families of jobs, indexed by $k_1, k_2, \ldots, k_K$, where $k_1 = 1$, and for any $i < K$, $k_{i+1} = k_i + 4\log\log m$. The family k_1 contains $m/\log m$ jobs, where each job has only one operation, the length of each job is $\sqrt{m}$, and the operations are all distinct. For $1 < i \le K$, family k_i is a bucket avoiding family $F(k_i, m, k_{i-1})$, where the length of each operation is $\sqrt{m}/2^{k_i}$. Hence, family k_i contains $m/\log m$ jobs, each job has 2^{k_i} operations, and the length of each job is $\sqrt{m}$.

Lemma 7.23 *Any legal schedule for the above jobs is of length at least $\frac{\sqrt{m}\log m}{32\log\log m}$.*

Proof. Fix any legal schedule, and assume that it is of length $c\sqrt{m}$, where $c < \frac{\log m}{32\log\log m}$. We shall derive a contradiction.

For each $k_i > 1$, we define a bucket configuration $B(m, k_i - 2)$ by induction relative to the assumed schedule. Each bucket configuration will be associated with $\sqrt{m}/8$ time steps of the schedule in which the jobs in family k_i are active, and the time steps associated with different bucket configurations will be disjoint. This will imply that the schedule must have a length of at least $\frac{\sqrt{m}}{8}\frac{\log m}{4\log\log m} = \sqrt{m}\log m/32\log\log m$.

Base case: For $k_1 = 1$, partition the schedule into $2c$ blocks of consecutive time steps, where each block is of length $\sqrt{m}/2$. As operations of family k_1 take $\sqrt{m}$ time units, each operation completely covers at least one block. As there are $m/\log m$ operations in family k_1, there is at least one block that is covered by at least $\frac{m}{2c\log m}$ operations. Take one such block arbitrarily and $m/(\log m)^2$ of the machines that cover it. The bucket associated with k_1 contains the $m - m/(\log m)^2$ other machines. Note that for any $i > 1$, a sequence of operations of a job from family k_i can be scheduled at the same time period in which the block corresponding to the k_1 bucket was scheduled only if all machines that have operations within the sequence are in the bucket.

Inductive step: Suppose we have already defined the bucket configuration for all k_j with $j \le i$. We shall now define the bucket configuration associated with k_{i+1}.

For each bucket in each bucket configuration up to k_i, mark on the schedule the time steps corresponding to the block that created each bucket. These time steps are disjoint (disjointness of these time steps is maintained by the inductive step). The total number of buckets that we have is $1 + \sum_{j=2}^{i} 2^{k_j - 2} \le 2^{k_i}$, and each bucket has $m - m/(\log m)^2$ operations. Hence, this can be regarded as (a part of) a bucket configuration $B(m, k_i)$. Recall that the family k_{i+1} was chosen as a bucket avoiding family $F(k_{i+1}, m, k_i)$. This implies that if the schedule is legal, at least half of the jobs of this family avoid the bucket configuration $B(m, k_i)$. Hence, at least a quarter of the operations of the family are performed at time steps that are not covered by previous locations of buckets. Call these operations *good*.

Partition now the time steps of the schedule into $2c2^{k_{i+1}}$ blocks, each of size $\sqrt{m}/2^{k_{i+1}+1}$. Remove any block that overlaps the location of a previous bucket. Each good operation (which is of length $\sqrt{m}/2^{k_{i+1}}$) completely covers one of the remaining blocks. As there are at least $m2^{k_{i+1}}/4\log m$ good operations, and each block can contain at most $m/2\log m$ good operations (this is the number of jobs involved), at least $2^{k_{i+1}}/4$ of the blocks have $m/16c\log m > m/(\log m)^2$ good operations in them. These blocks make the bucket configuration associated with k_{i+1}, where the machines going into each bucket are those machines that do not cover the respective block. The total number of time steps associated with these buckets is $(2^{k_{i+1}}/4)(\sqrt{m}/2^{k_{i+1}+1}) = \sqrt{m}/8$. By construction, the time steps associated with these buckets are disjoint from those associated with previous buckets. □

To complete the proof of Theorem 7.22, observe that by construction, jobs are acyclic. The dilation (length of the longest job) is $\sqrt{m}$. The total number of jobs is $n = Km/\log m \simeq m/4\log\log m$. The average load per machine is $n\sqrt{m}/m < \sqrt{m}/4\log\log m$. By condition 1 in Definition 7.20 it follows that the congestion is at most $\sqrt{m}/2\log\log m$. □

Remark: For the particular instance of JSS that we construct, Lemma 7.23 is best possible up to constant factors. Each family of jobs in our instance can be scheduled to require $O(\sqrt{m})$ time (this follows from the main result of [LMR94], as all operations within a family are of the same length), and there are $O(\log m/\log\log m)$ families. The constant factors can be improved by tightening parameters of the construction (e.g., making $k_{i+1} = k_i + 3\log\log m$, saying that a job avoids a bucket configuration if almost all of its operations remain uncovered) and tightening the analysis, giving a lower bound of $\sqrt{m}\log m/6\log\log m$, or $D\log D/3\log\log D$.

7.4 Conclusion

In this chapter, we presented nearly matching upper and lower bounds for several classes of acyclic job shop scheduling problems. Many questions remain open. Here are our favorite ones:

- **General JSS:** For this case, we have not been able to improve the upper bound of $L = O(lb(\log lb)^2/(\log\log lb)^2)$. How tight is this upper bound?

- **Preemptive JSS:** For acyclic JSS with preemption we show that $L = O(lb\log\log lb)$. We do not know if this is best possible. Is $L = \Theta(lb)$? For general JSS with preemption we were unable to improve over the previously known upper bound of $L = O(lb\log lb/\log\log lb)$, nor provide any evidence that $L = \Theta(lb)$ is unachievable.

- **Flow shop scheduling:** An important special case of acyclic JSS is flow shop scheduling, where machines are ordered, and within each job, the order in which machines are requested respects the order of the machines. Previous to our work, the best upper bound known for flow shop scheduling was similar to that of general JSS, namely $L = O(lb(\log lb)^2/(\log\log lb)^2)$. Clearly, our upper bound of $L = O(lb\log lb\log\log lb)$ for acyclic JSS applies also to the special case of flow shop scheduling. However, this does not hold for our lower bound of $L = \Omega(lb\log lb/\log\log lb)$. Can the upper bound for flow shop scheduling be improved significantly?

Chapter 8

Coloring Non-Uniform Hypergraphs

The Lovász Local Lemma is a sieve method to prove the existence of certain structures with certain prescribed properties. In most of its applications the Lovász Local Lemma does not supply a polynomial-time algorithm for finding these structures. Beck was the first who gave a method of converting some of these existence proofs into efficient algorithmic procedures, at the cost of losing a little in the estimates. He applied his technique to the symmetric form of the Lovász Local Lemma and, in particular, to the problem of 2-coloring uniform hypergraphs.

In this chapter, we investigate the general form of the Lovász Local Lemma. Our main result is a randomized algorithm for 2-coloring *non-uniform* hypergraphs that runs in expected *linear* time. Even for uniform hypergraphs no algorithm with such a runtime bound was previously known, and no polynomial-time algorithm was known at all for the class of non-uniform hypergraphs we will consider in this chapter. Our algorithm and its analysis provide a novel approach to the general Lovász Local Lemma that may be of independent interest. We also show how to extend our result to the c-coloring problem.

8.1 Introduction

The probabilistic method is used to prove the existence of objects with desirable properties by showing that a randomly chosen object from an appropriate probability distribution has the desired properties with positive probability. In most applications this probability is not only positive, but is actually high and frequently tends to 1 as the parameters of the problem tend to infinity. In such cases, the proof usually supplies an efficient randomized algorithm for producing a structure of the desired type.

There are, however, certain examples, where one can prove the existence of the required combinatorial structure by probabilistic arguments that deal with rare events; events that hold with positive probability which is exponentially small in the size of the input. This happens often when using the Lovász Local Lemma.

Many applications of the Lovász Local Lemma can be found in the literature (see Section 3.6). In all of these applications the Lovász Local Lemma only guarantees an extremely small (though positive) probability that no bad event holds. To turn proofs using the Lovász Local Lemma into efficient algorithms, even random ones, proved to be difficult for many of these applications. In a breakthrough paper [Bec91], Beck presented a method of converting some applications of the Lovász Local Lemma into polynomial-time algorithms (with some sacrifices made with regards to the constants in the original application). Alon [Alo91] provided a parallel variant of the algorithm and simplified the arguments used. Their methods were further generalized by Leighton, Maggs and Richa [LMR99] and Molloy and Reed [MR98] to yield efficient algorithms for a number of other applications of the Lovász Local Lemma. However, all of these approaches can be applied only to the *symmetric* form of the Lovász Local Lemma, in which a very regular structure of the events under consideration is required.

Molloy and Reed [MR98] also found methods that could possibly be applied to problems that require the general Lovász Local Lemma, but as it was pointed out by the authors, they may require the proof of some (possibly difficult) concentration-like properties for each problem under consideration.

8.1.1 New Results

In this chapter, we present a novel method for turning some applications of the general LLL into efficient algorithms. We apply it to the problem of 2-coloring non-uniform hypergraphs. No polynomial-time algorithm has been found so far for this problem.

To start with, we first require some notation. Given a hypergraph $H = (V, E)$, let the *(1-)neighborhood* $N(e)$ of an edge $e \in E$ be defined as $N(e) = \{e' \in E \setminus \{e\} \mid e \cap e' \neq \emptyset\}$. Similarly, for $\mathcal{E} \subseteq E$ we define $N(\mathcal{E}) = \bigcup_{e\in\mathcal{E}} N(e) \setminus \mathcal{E}$. For any $e \in E$, let $|e|$ denote the *size* of edge e (i.e., the number of nodes e contains). A hypergraph $H = (V, E)$ is *c-colorable* if c colors suffice to color its nodes in such a way that no hyperedge in E is monochromatic (i.e., has only nodes of a single color).

The following result due to Erdős and Lovász [EL75] follows easily from the Lovász Local Lemma (see Section 3.6.1).

Theorem 8.1 [EL75] *Consider any hypergraph $H = (V, E)$ in which every edge has at least $k \geq 2$ nodes and no edge intersects more than d other edges. If $\mathrm{e}\,(d + 1) \leq 2^{k-1}$, then H is 2-colorable.*

We note that Radhakrishnan and Srinivasan [RS98] were able to improve this theorem. They showed that if no edge intersects more than $0.17 \cdot \sqrt{k/\ln k} \cdot 2^k$ edges, then for a sufficiently large k, hypergraph H is 2-colorable. However, their result is also non-constructive.

Efficient algorithms for finding such a 2-coloring were given by Beck [Bec91] and later by Alon [Alo91]. They show the following result (see Section 3.6.2).

Theorem 8.2 [Alo91, Bec91] *There is a positive constant $c > 1$, such that if $\mathrm{e}\,(d+1) \leq 2^{k/c}$, then any hypergraph H in which every edge contains at least k nodes and no edge intersects more than d other edges can be 2-colored in polynomial time.*

Theorems 8.1 and 8.2 are interesting mostly for uniform hypergraphs. (Indeed, the way that one can color the nodes of a hypergraph H in which every edge contains at least k nodes is by reducing each edge to an arbitrarily chosen set of k of its nodes and solving the problem for uniform hypergraphs.) However, they provide no reasonable bounds for non-uniform hypergraphs in which the edge sizes can vary arbitrarily (especially, when some edges might be of very small size and some other edges of very large size).

One can generalize Theorem 8.1 to the following result for non-uniform hypergraphs. We state it here in a form that makes it easily comparable with our constructive result, Theorem 8.4.

Theorem 8.3 *Let $\mathcal{H}$ be a hypergraph with edges $e_1, \ldots, e_m$. For every i, $1 \leq i \leq m$, let A_i be the event that e_i is monochromatic. Consider the random experiment of selecting a color for* *each node independently and uniformly at random (i.u.r.). Then* $\Pr[A_i] = 2^{1-|e_i|}$*. Assume that there exist* $x_j = 2^{-\delta_j\,|e_j|}$ *(for a suitably chosen* $0 < \delta_j < 1$*) for all* $1 \leq j \leq m$ *with*

$$\Pr[A_i] \leq x_i \prod_{e_j \in N(e_i)} (1 - x_j) \; .$$

Then $\mathcal{H}$ *is* 2*-colorable.*

None of the techniques developed so far can be used to construct an efficient algorithm for finding a 2-coloring for this class of hypergraphs. We will present a randomized algorithm that finds a 2-coloring in an efficient way for the following hypergraphs.

Theorem 8.4 *There exist constants* K, Δ, $\mathcal{E} > 0$ *such that for any* $k \geq K$, $0 < \delta \leq \Delta$, *and* $0 < \epsilon \leq \mathcal{E}$ *it holds:*

Let $\mathcal{H}$ *be a hypergraph with edges* $e_1, \ldots, e_m$*, in which every edge is of size at least* k*. For every* i, $1 \leq i \leq m$*, let* A_i *be the event that* e_i *is monochromatic. Consider the random experiment of selecting a color for each node i.u.r. Assume that for* $x_j = 2^{-\delta\,|e_j|}$ *for all* $1 \leq j \leq m$ *it holds that*

$$(\Pr[A_i])^{\epsilon} \leq x_i \prod_{e_j \in N(e_i)} (1 - x_j) \; .$$

Then there is a randomized algorithm that finds a 2*-coloring of* $\mathcal{H}$ *in expected time linear in* $\sum_i |e_i|$.

Remark 8.5 *In this paper we did not make any attempt to optimize the values of* K, Δ *and* $\mathcal{E}$*. In our proofs we require* $\epsilon \leq 1/24$, $\delta \leq \epsilon^2/12$, *and* $k \geq 1/\delta$ *to obtain a polynomial-time algorithm. For a linear-time algorithm we need* $\epsilon \leq 1/24^2$ *and* $\delta \leq \epsilon^{3/2}/12$*. If at least 4 colors are available then we show in Section 8.5 that the hyperedges are allowed to be of arbitrary size, i.e.,* $K = 1$.

Observe that Theorem 8.4 contains as a special case Theorem 8.2. Thus it is its generalization to non-uniform hypergraphs. Most remarkable about this theorem is that we are able to construct an algorithm that runs in expected *linear time*. No such fast algorithm was known before even for uniform hypergraphs.

Theorem 8.4 implies the following result, answering an open question of J. Beck.

Corollary 8.6 *Let* $H = (V, E)$ *be a hypergraph. There exist positive constants* c_1, c_2, c_3 *so that if every edge in* H *is of size at least* c_1 *and for every integer* k *no edge* $e \in E$ *intersects more than* $c_2 \cdot |e| \cdot 2^{c_3 \cdot k}$ *edges of size at most* k*, then one can find in polynomial time a* 2*-coloring of* H*, w.h.p.*

The underlying structure of our 2-coloring algorithm is similar to that of Beck [Bec91] and Alon [Alo91] for uniform hypergraphs:

Step 1: Color each node of H i.u.r.
Step 2: Select nodes that require a recoloring.
Step 3: Recolor the selected nodes.

However, our algorithm differs from Beck's and Alon's approach by the fact that it uses a much more restrictive selection of the nodes that need a recoloring and allows edges to be reduced to a fraction of their nodes if necessary. These two features enable us to prove that each set of selected edges covers a small (at most logarithmic) number of nodes, with high probability. This will allow us to find in polynomial time a 2-coloring even for non-uniform hypergraphs.

As in the paper due to Alon [Alo91] (see the proof of Theorem 3.79), our approach to bound the runtime of the algorithm is to count certain structures that could "witness" a bad behavior of the algorithm. However, his method of looking at the 2- and 3-neighborhood of bad events seems not to be applicable to non-uniform hypergraphs. Our structures will allow to consider the 1-neighborhood of bad events, which is vital to achieve our results.

Our techniques for the 2-coloring problem easily extend to the problem of c-coloring non-uniform hypergraphs. Many other applications also seem to be in reach:

Consider any discrete, finite application of the general Lovász Local Lemma. Let $\mathcal{A} = \{A_1, \ldots, A_n\}$ be its set of events and $\mathcal{T} = \{t_1, \ldots, t_m\}$ be the set of independent trials the events are based on. Each trial has a set of possible outcomes which may be viewed as its set of colors. Suppose that A_i only depends on a set of trials $T_i \subseteq \mathcal{T}$. Then the problem of making all events in $\mathcal{A}$ false can be interpreted as a generalized form of the hypergraph coloring problem, where the T_i represent the hyperedges and $\mathcal{T}$ represents the set of nodes.

We will look at some of these general problems in the next chapter.

In this chapter, we shall only present randomized, sequential algorithms. We remark, however, that our algorithms for the 2-coloring and c-coloring problem can be transformed into deterministic polynomial-time algorithms using the techniques in [AGHP92, EGL$^+$98, NN93] (see also the discussion in [Alo91, p.371] and in [RS98, Section 3]). Our method can also be implemented as a parallel $\mathcal{NC}$ algorithm. Since these two results can be obtained using standard methods, we leave their details to the reader.

8.1.2 Organization of the Chapter

In the next section we will present a polynomial-time algorithm for the 2-coloring problem. Section 8.3 contains the analysis of the algorithm. In Section 8.4 we show how to transform the polynomial-time algorithm into a linear-time algorithm. Afterwards, we present in Section 8.5 generalizations to the c-coloring problem. Section 8.6 contains some technical parts of our calculations. We conclude the chapter with some open problems.

8.2 Description of the Algorithm

We first present a polynomial-time algorithm for the 2-coloring problem. Consider any hypergraph $H = (V, E)$ that fulfills the conditions in Theorem 8.4, and let ϵ be chosen as in Remark 8.5. Our algorithm consists of three steps.

Step 1: Color all the nodes of H by choosing for each node one out of two possible colors independently and uniformly at random.

Before we describe Step 2, we first introduce some notation and provide the ideas behind that step. An edge e is called *bad* if more than $(1 - 2\,\epsilon)\,|e|$ of its nodes have the same color after Step 1. Otherwise the edge is called *good*. That is, a good edge e has at least $2\,\epsilon\,|e|$ nodes of each color.

Clearly, after Step 1 there might be many monochromatic edges left in H. In this case we have to recolor certain nodes in these edges. Similar to [Alo91, Bec91], we shall not only recolor the nodes in the monochromatic edges, but also those in the bad edges. The aim of Step 2.1 is to find a partition of the bad edges into node-disjoint groups (i.e., two bad edges from different groups should be node-disjoint). The key feature of our partitioning procedure is that we do not consider *all* the nodes covered by the bad edges. Instead, in the course of the algorithm we shall frequently *reduce* the edges, that is, we shall modify the edges by removing some subset of their nodes (see Figure 8.1).

Figure 8.1: *Reduction* of an edge e to edge e^*. Observe that if e^* is non-monochromatic, then so is e.

Step 2: Perform the following two substeps Step 2.1 and Step 2.2.

Step 2.1:
set $R = E$ // R denotes the set of remaining edges
repeat
  choose any bad edge e in R
  call **Build_1-Component**(e)
until there are no bad edges left in R

Algorithm Build_1-Component(e_0)**:**
set $i = 0$ and $E_0 = \{e_0\}$
repeat
  set $E_{i+1} = \emptyset$
  for all edges $e \in E_i$ do:
    for all edges $e' \in N(e) \cap R$ do: // e' has not yet been considered
      set $e'' = e' \cap (\bigcup_{j=0}^{i+1} \bigcup_{e^* \in E_j} e^*)$ // e'' is the set of nodes in e' already covered
      if $|e''| \geq \epsilon\, |e'|$ **then**
        remove e' from R, reduce it to e'', and add it to E_{i+1}
      else
        if e' is bad **then** remove it from R and add it to E_{i+1} $(*)$
  set $i = i + 1$
until $E_i = \emptyset$

After performing Step 2.1, one could (even in polynomial time, as we will show) recolor the nodes within each group of bad edges constructed in Build_1-Component independently of other groups such that all bad edges would become non-monochromatic. This, however, could make some of the good edges monochromatic. Therefore, in Step 2.2 we consider the good edges that are monochromatic after removing those of their nodes that belong to any group chosen in Step 2.1. The aim of Step 2.2 is to either *reduce* each such edge to the nodes covered

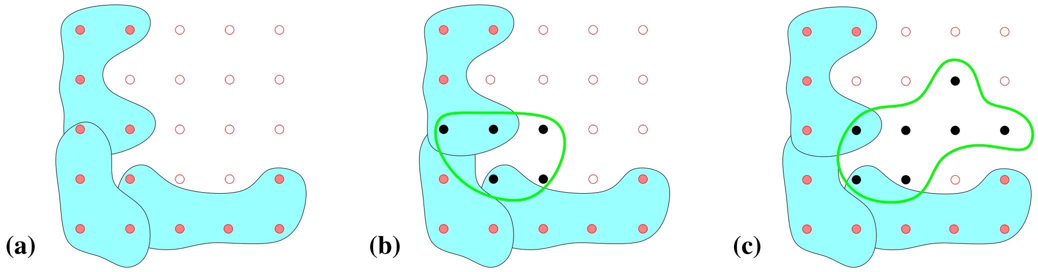

Figure 8.2: Illustration to Step 2.1. (a) An example of three edges defining set $\bigcup_{j=0}^{i+1} E_j$. (b) The case when a new edge e' intersects the edges in $\bigcup_{j=0}^{i+1} E_j$ in at least $\epsilon\,|e'|$ of its nodes. In this case e' is reduced to the three nodes in $\bigcup_{j=0}^{i+1} E_j$. In Step 3 of the algorithm we shall recolor these three nodes to ensure that e' will not be monochromatic. (c) The case when a new edge e' has a very small intersection with the edges in $\bigcup_{j=0}^{i+1} E_j$. In this case we do not reduce e'. If e' is good, then the nodes outside $\bigcup_{j=0}^{i+1} E_j$ will ensure that e' will remain monochromatic. However, if e' is bad, then we add it completely to the set E_{i+1}.

by bad edges from one group chosen in Step 2.1, or to *combine* into one group all the groups covering the given edge. As we will see, the nodes covered by the groups constructed in Step 2.2 can be recolored independently so that now every edge is ensured to be non-monochromatic at the end.

Let the set $\bigcup_{i\geq 0} E_i$ built in the algorithm Build_1-Component be called a *1-component*. For each 1-component C, let B_C denote the set containing the edge e_0 in C and the edges that were added to C in $(*)$ because they were bad. These edges are called the *core edges* of C. They play an important role in the recoloring process in Step 3 because, as we shall show, our choice of ϵ ensures that it is sufficient to recolor only the nodes covered by the core edges in order to get a non-monochromatic coloring for all edges in C.

Define $\mathcal{C}$ to be the set of all 1-components. In the following, let $E^* = E \setminus (\bigcup_{C\in\mathcal{C}} C)$ denote the set of all edges not assigned to any of the 1-components. An edge e in E^* is called *dangerous* if at least $2\,\epsilon\,|e|$ of its nodes are covered by core edges*. The second substep works as follows.

Step 2.2:
set $R = \mathcal{C}$ // R denotes the set of remaining 1-components
repeat
  choose any 1-component C in R
  call **Build_2-Component**(C)
until R is empty

*This definition is oriented towards our analysis presented in the next section. For the purpose of the algorithm it would be enough to consider only those edges for which all nodes not covered by the core edges are of the same color.

Algorithm Build_2-Component(C)**:**

set $i = 0$ and $E_0 = \{e_0\}$, where e_0 is any edge in B_C
repeat
  set $E_{i+1} = (N(E_i) \cap B_C) \setminus (\bigcup_{j=0}^{i} E_j)$ // initial 1,2-neighborhood of E_i
    for all edges $e' \in N(E_i) \cap E^*$ do: // e' has not yet been considered
      set $e'' = e' \cap (\bigcup_{b \in B_C} b)$ // e'' is the set of nodes in e' already covered
      if $|e''| \geq \epsilon\,|e'|$
        then reduce e' to e'' and add it to C
        else
          if e' is dangerous **then** $(**)$
            for all 1-components C' in R that overlap with e' do:
              add to E_{i+1} all edges in $B_{C'}$ that intersect e'
              remove C' from R, add C' to C and $B_{C'}$ to B_C
            reduce e' to the nodes covered by $\bigcup_{b \in B_C} b$ and add it to C
      remove e' from E^*
  set $i = i + 1$
until $E_i = \emptyset$

Let the set C built in the algorithm Build_2-Component be called a *2-component*. For each 2-component C, let B_C denote the union of the sets $B_{C'}$ of the 1-components C' it consists of and let V_C be the set of nodes covered by the edges in B_C.

Remark 8.7 *Let us notice five important properties that we shall frequently use in our analysis and which follow immediately from our construction:*

(1) For every 2-component C, all edges in B_C are bad.
(2) For every 2-component C, it holds that $\bigcup_{e \in C} e = \bigcup_{e \in B_C} e$.
(3) For every edge e in the hypergraph H, if it is contained as a reduced edge e' in a 2-component C, then $|e'| \geq \epsilon\,|e|$.
(4) For every 2-component C, if $B_C = \{e_1, \ldots, e_\ell\}$, then there exists a partition of V_C into $W_1, \ldots, W_\ell$ such that for every i, $1 \leq i \leq \ell$, it holds that $W_i \subseteq e_i$ and $|W_i| \geq (1-\epsilon)\,|e_i|$.
(5) If an edge e is bad (after completing Step 1), then there exists a 2-component which contains either e or the reduced edge of e.

Step 3: Find a coloring for each 2-component so that all of its edges are non-monochromatic.

In order to perform Step 3 efficiently, we use the following properties.

Lemma 8.8

(1) For every 2-component C, there is a coloring of the nodes in V_C such that all edges in C are non-monochromatic.
(2) No edge that is not assigned to any 2-component can become monochromatic by the recoloring process in Step 3.

Proof. We first prove (1). By Remark 8.7 (3), every edge in C has at least $\epsilon\,|e|$ of its nodes in V_C. Hence, the probability bound in Theorem 8.4 together with the LLL imply that there is a 2-coloring for the nodes in V_C so that every edge in C is non-monochromatic, no matter what the color of its nodes outside of V_C is.

In order to prove (2), let us consider any edge e that was not assigned to any 2-component. This means that it was neither bad nor dangerous. That is, it has at least $2\,\epsilon\,|e|$ nodes of each color and less than $2\,\epsilon\,|e|$ of its nodes are covered by core edges. Hence, the nodes of e not covered by the core edges are still non-monochromatic, and therefore e cannot become monochromatic by the recoloring process in Step 3. □

Lemma 8.8 and the fact that the node sets covered by distinct 2-components are disjoint imply the following vital property.

Corollary 8.9 *The 2-components can be recolored independently of each other in order to obtain a correct 2-coloring.*

We will prove in the next section (Theorem 8.10) that the number of nodes to be recolored in each 2-component is $O(\log m)$, w.h.p. Together with property (1) of Lemma 8.8, this allows us to find a proper coloring for each 2-component via exhaustive search in polynomial time. Hence, Corollary 8.9 implies that Step 3 can be performed in polynomial time. This results in a polynomial-time algorithm for the 2-coloring problem. To obtain a linear-time algorithm, we run in two phases. The first phase consists of Steps 1 and 2 above and the second phase is based on Steps 1 to 3 above, applied independently to each of the 2-components resulting from Phase 1. We shall describe this procedure in detail in Section 8.4. This will establish Theorem 8.4.

8.3 Analysis of the Algorithm

In this section we prove the following result.

Theorem 8.10 *If ϵ, δ, and k in Theorem 8.4 are set as $\epsilon = 1/24$, $\delta = \epsilon^2/12$, and $k = 1/\delta$, then with probability at least $1 - \frac{1}{m}$ the size of every 2-component is $O(\log m)$.*

In order to prove this theorem, we consider all possible structures of a certain kind that could witness a large 2-component. These structures are described in Definition 8.12. For this we need the following definition.

Definition 8.11 *Given a hypergraph $H = (V, E)$ and a set of edges $\mathcal{E} \subseteq E$, let*

- *the k-*neighborhood *of $\mathcal{E}$ be defined as*

$$N_k(\mathcal{E}) = \begin{cases} \mathcal{E} & : \quad k = 0 \\ N(\mathcal{E}) & : \quad k = 1 \\ N(N_{k-1}(\mathcal{E})) \setminus N_{k-2}(\mathcal{E}) & : \quad k > 1 \end{cases}$$

- *and the k,ℓ-neighborhood of $\mathcal{E}$ be defined as $N_{k,\ell}(\mathcal{E}) = \cup_{i=k}^{\ell} N_i(\mathcal{E})$.*

Definition 8.12 *Consider any hypergraph $H = (V, E)$.*
A sequence $\mathcal{W} = \langle B_0, B_1, \ldots, B_d \rangle$ of edge sets $B_i \subseteq E$ is called a witness structure *of depth d if*

- *B_0 contains a single edge and*
- *all edges in B_{i+1} are in the 1,2-neighborhood of B_i.*

A witness structure is called a 2-component witness *if*

(1) for every edge $e \in \bigcup_{i=0}^{d} B_i$ a subset e' of its nodes of size at least $(1-\epsilon)\,|e|$ can be chosen such that all e' are disjoint and
(2) for every $i \in \{0,\ldots,d-1\}$ the edges in B_{i+1} that are in the 2-neighborhood of B_i can be partitioned into sets $S_1,\ldots,S_r$ for some r, such that for every S_j there is a distinct edge $e_j \in N(B_i) \setminus B_{i+1}$ with $|e_j \cap (\bigcup_{e \in S_j} e')| \geq \epsilon\,|e_j|$ (note that e' is the part of e defined in (1)).

Furthermore, a witness structure is called valid *if all edges in $\bigcup_{i=0}^{d} B_i$ are bad.*
Let $V_{\mathcal{W},i} = \bigcup_{e \in B_i} e$ denote the set of nodes covered by B_i and let $V_{\mathcal{W}} = \bigcup_{i=0}^{d} V_{\mathcal{W},i}$ denote the set of all nodes covered by $\mathcal{W}$.

We introduce these structures, because our aim will be to bound the expected number of valid 2-component witnesses of a certain size, rather than trying to bound the expected number of 2-components of a certain size constructed by the algorithm. This is important, since the latter method invokes dependencies that seem to be extremely difficult to handle, whereas the former method is purely combinatorial in that it is not based on a certain selection process. The next lemma enables us to switch to these witness structures.

Lemma 8.13 *For any 2-component C constructed by the algorithm, there is a valid 2-component witness that contains all edges in B_C.*

Proof. In order to construct a suitable witness structure for C, let us simply choose $B_i = E_i$ in the Build_2-Component algorithm for all $i \geq 0$. It is easy to check that this construction fulfills all requirements of a witness structure. Furthermore, by Remark 8.7, our algorithm ensures that disjoint node sets can be assigned to the edges in B_C so that every edge $e \in B_C$ is assigned to at least $(1-\epsilon)\,|e|$ of its nodes. Moreover, line $(**)$ in Step 2.2 ensures that Property (2) of Definition 8.12 is fulfilled. Hence, the witness structure constructed above is a 2-component witness. It is also valid since our algorithm requires all edges in B_C to be bad. □

Lemma 8.13 implies that if there is no valid 2-component witness over an edge set B then B cannot form a 2-component. The proof of Theorem 8.10 therefore follows directly from the following lemma.

Lemma 8.14 *There exists a positive constant γ such that*

$$\mathrm{E}\left[|\{\textit{valid 2-component witnesses } \mathcal{W} : |V_{\mathcal{W}}| \geq \gamma \cdot \log m\}|\right] \leq \frac{1}{m} .$$

Before we prove the lemma, let us first introduce some notation which will be frequently used in the subsequent sections.

Definition 8.15 *Given any edge set $\mathcal{E}$, let $V_{\mathcal{E}}$ denote the set of all nodes covered by edges in $\mathcal{E}$. $\mathcal{E}$ is called a* core witness *if for each edge $e \in \mathcal{E}$ a subset e' of its nodes of size at least $(1-\epsilon)\,|e|$ can be chosen such that all e' are disjoint. Furthermore, $\mathcal{E}$ is called* valid *if all edges in $\mathcal{E}$ are bad. An edge set $\mathcal{F}$ is called a* 1,2-core witness *of $\mathcal{E}$ if $\mathcal{F}$ is a core witness and $\mathcal{F}$ is related to $\mathcal{E}$ as B_{i+1} is related to B_i in Definition 8.12 (2). Finally, let $\hat{\nu}_{\mathcal{E}}$ denote the maximum number of nodes a valid 1,2-core witness of $\mathcal{E}$ can cover.*

Lemma 8.14 now follows from the following three propositions.

Proposition 8.16 *There is a constant $c > 0$ so that for every edge e of size larger than $c \log m$ it holds*

$$\Pr[e \text{ is bad}] \leq \frac{1}{2m^2} .$$

In particular,

$$\Pr[\text{there is a valid 2-component witness } \mathcal{W} \text{ with } |V_{\mathcal{W},0}| \geq c \log m] \leq \frac{1}{2m} .$$

The proof of this proposition is obvious. Recall that, for each edge of a 2-component witness, we can select at least a $(1-\epsilon)$-fraction of its nodes that is disjoint from the node sets assigned to other edges. Therefore, we can assume that the probability of an edge to be bad is "independent" of other edges by considering only the subset of nodes chosen for it and assuming the worst possible case for the remaining nodes (cf. Claim 8.23). Furthermore, Property (2) of Definition 8.12 allows us to assume that also the probability of an edge to be dangerous is "independent" (in the above sense) of other edges. These are key properties which will enable us to obtain the following result.

Proposition 8.17 *There are constants $0 < \alpha < 1$ and $0 < \beta \leq 1/2$ so that for every core witness $\mathcal{E}$ we have $\mathrm{E}[\hat{\nu}_{\mathcal{E}}] \leq \beta |V_{\mathcal{E}}|$ and for every $\nu \geq |V_{\mathcal{E}}|/2$ it holds that $\Pr[\hat{\nu}_{\mathcal{E}} \geq \nu] \leq \alpha^{\nu}$.*

Proposition 8.17 will be proven in Section 8.3.4 after some preparations in Sections 8.3.1–9.2.5. The proof of the next proposition is given in Section 8.6.

Proposition 8.18 *Let α and β be positive constants with $0 < \alpha < 1$ and $0 < \beta \leq \frac{1}{2}$. Let $X_0, X_1, \ldots,$ be any sequence of non-negative integer random variables satisfying the following four conditions:*

- $X_0 = \lambda$,
- $\mathrm{E}[X_{i+1} \mid X_0, X_1, \ldots, X_i] \leq \beta \cdot X_i$ *for every* $i \geq 0$,
- *if* $X_i = 0$ *then* $X_{i+1} = 0$, *and*
- $\Pr[X_{i+1} \geq t \mid X_0, X_1, \ldots, X_i] \leq \alpha^t$ *for every* $i \geq 0$ *and every* $t \geq \frac{1}{2} \cdot X_i$.

Then $\mathrm{E}\left[\sum_{j \geq 0} X_j\right] \leq 2 \cdot \lambda$, *and there is a positive constant c such that for every $s > 0$*

$$\Pr\left[\sum_{j \geq 0} X_j \geq c \cdot (\lambda + \log s)\right] \leq \frac{1}{s} .$$

Proof of Lemma 8.14: From Proposition 8.16 it follows that

$$\mathrm{E}[|\{\text{valid 2-component witnesses } \mathcal{W}\text{: } |V_{\mathcal{W},0}| \geq c_1 \log m\}|] \leq \frac{1}{2m} ,$$

and from Propositions 8.17 and 8.18 we get that

$$\mathrm{E}[|\{\text{valid 2-component witnesses } \mathcal{W}\text{: } |V_{\mathcal{W}}| \geq c_2(|V_{\mathcal{W},0}| + \log m)\}|] \leq \frac{1}{2m} ,$$

for some positive constants c_1 and c_2. Hence, for $\gamma \geq c_2(c_1+1)$,

$$\begin{aligned}
&\mathrm{E}\left[|\{\text{valid 2-component witnesses } \mathcal{W}\colon |V_{\mathcal{W}}| \geq \gamma \cdot \log m\}|\right] \\
&\quad = \mathrm{E}[|\{\text{valid 2-component witnesses } \mathcal{W}\colon |V_{\mathcal{W},0}| \geq c_1 \log m \text{ and } |V_{\mathcal{W}}| \geq \gamma \cdot \log m\}|] + \\
&\qquad \mathrm{E}[|\{\text{valid 2-component witnesses } \mathcal{W}\colon |V_{\mathcal{W},0}| < c_1 \log m \text{ and } |V_{\mathcal{W}}| \geq \gamma \cdot \log m\}|] \\
&\quad \leq \mathrm{E}[|\{\text{valid 2-component witnesses } \mathcal{W}\colon |V_{\mathcal{W},0}| \geq c_1 \log m\}|] + \\
&\qquad \mathrm{E}[|\{\text{valid 2-component witnesses } \mathcal{W}\colon |V_{\mathcal{W}}| \geq c_2(|V_{\mathcal{W},0}| + \log m)\}|] \\
&\quad \leq \frac{1}{2m} + \frac{1}{2m} = \frac{1}{m}\ .
\end{aligned}$$

This completes the proof of Lemma 8.14. □

8.3.1 A Technical Lemma

In this section we present a lemma that will be essential for estimating the expected number of valid 1,2-core witnesses that cover a given number of nodes.

Suppose that for every edge $e \in E$ we have an integer random variable Λ_e which we call its *contribution*. For any set of edges $\mathcal{F}$, let the *contribution* $\Lambda_{\mathcal{F}}$ of $\mathcal{F}$ be defined as the sum of the contributions of all edges in $\mathcal{F}$. Let

$$S_\lambda^{\mathcal{E}} = \{\mathcal{F} \subseteq N(\mathcal{E}) \ : \ \Lambda_{\mathcal{F}} = \lambda\}\ .$$

The next lemma provides an estimation for $\mathrm{E}[|S_\lambda^{\mathcal{E}}|]$.

Lemma 8.19 *Let c and γ be arbitrary positive constants with $\gamma \leq 1/288$. Furthermore, let $\mathcal{E}$ be any set of edges and $(\Lambda_e)_{e \in N(\mathcal{E})}$ be any sequence of contributions with the property that*

(1) the contribution of any edge in $N(\mathcal{E})$ is either 0 or at least $1/\gamma$,

(2) $|\{e \in N(\mathcal{E}) : \Pr[1 \leq \Lambda_e \leq k] > 0\}| \leq c \cdot 2^{\gamma \cdot k}$, and

(3) there is a $\phi \geq 1/6$ so that for every edge $e \in N(\mathcal{E})$, $\Pr[\Lambda_e = k] \leq 2^{-\phi \cdot k}$ independently of other events of positive contribution.

Then

$$\mathrm{E}[|S_\lambda^{\mathcal{E}}|] \leq 2^{-\phi\lambda/2} \cdot 2^{c/48}\ .$$

Proof. Let $m_\lambda^{\mathcal{E}}$ denote the number of all possible candidates for sets in $S_\lambda^{\mathcal{E}}$. We first bound $m_\lambda^{\mathcal{E}}$. For every set of edges $\{e_1, \ldots, e_r\} \in N(\mathcal{E})$, a sequence $\langle s_1, \ldots, s_\lambda \rangle$ is the *contribution-characterization* of $\{e_1, \ldots, e_r\}$ if $s_k = |\{1 \leq j \leq r : \Lambda_{e_j} = k\}|$. Since there are at most $c \cdot 2^{\gamma \cdot k}$ edges of contribution k in $N(\mathcal{E})$, there can be at most

$$\prod_{k=1}^{\lambda} \binom{c \cdot 2^{\gamma \cdot k}}{s_k}$$

sets $\mathcal{F} \subseteq N(\mathcal{E})$ that fulfill a prescribed contribution-characterization $\langle s_1, \ldots, s_\lambda \rangle$.

Let $\mathcal{P}(\lambda) = \{\langle s_1, \ldots, s_\lambda \rangle$: each s_j is a non-negative integer and $\sum_{j=1}^{\lambda} j \cdot s_j = \lambda\}$. Furthermore, let $\mathcal{P}(\lambda;\kappa) = \{\langle s_1, \ldots, s_\lambda \rangle \in \mathcal{P}(\lambda) : s_1 = \cdots s_{\kappa-1} = 0\}$. By our discussion above we obtain

$$\begin{aligned}
m_\lambda^{\mathcal{E}} &\leq \sum_{\langle s_1,\ldots,s_\lambda \rangle \in \mathcal{P}(\lambda;1/\gamma)} \prod_{k=1}^{\lambda} \binom{c \cdot 2^{\gamma \cdot k}}{s_k} \\
&\leq \sum_{\langle s_1,\ldots,s_\lambda \rangle \in \mathcal{P}(\lambda;1/\gamma)} \prod_{k \geq 1/\gamma,\, s_k > 0} \left(\frac{\mathrm{e} \cdot 2^{\gamma \cdot k} \cdot c}{s_k}\right)^{s_k}\ .
\end{aligned}$$

The bound given in the lemma for the probability that an edge has a contribution of k yields

$$\begin{aligned}
\mathrm{E}[|S_\lambda^{\mathcal{E}}|] &\leq \sum_{\langle s_1,\ldots,s_\lambda\rangle \in \mathcal{P}(\lambda;1/\gamma)} \prod_{k\geq 1/\gamma,\, s_k>0} \left(\frac{\mathrm{e}\cdot 2^{\gamma\cdot k}\cdot c}{s_k}\cdot 2^{-\phi\cdot k}\right)^{s_k} \\
&= 2^{-2\phi\cdot\lambda/3} \sum_{\langle s_1,\ldots,s_\lambda\rangle \in \mathcal{P}(\lambda;1/\gamma)} \prod_{k\geq 1/\gamma,\, s_k>0} \left(\frac{\mathrm{e}\cdot 2^{\gamma\cdot k}\cdot c}{s_k\cdot 2^{\phi\cdot k/3}}\right)^{s_k} .
\end{aligned}$$

It is easy to check that, for any $s, n > 0$, $(\frac{n}{s})^s$ is maximal for $s = n/\mathrm{e}$. Hence, for any $\gamma \leq 1/72$,

$$\left(\frac{\mathrm{e}\, 2^{\gamma\cdot k}\cdot c}{s_k\cdot 2^{\phi\cdot k/3}}\right)^{s_k} \leq \mathrm{e}^{c\cdot\frac{2^{\gamma\cdot k}}{2^{\phi\cdot k/3}}} \leq \mathrm{e}^{c\cdot 2^{-\phi\cdot k/4}} .$$

Furthermore, for $\gamma \leq 1/288$ and $\phi \geq 1/6$ we have

$$\sum_{k\geq 1/\gamma} 2^{-\phi\cdot k/4} \leq 2^{-1/24\gamma}\sum_{k\geq 0} 2^{-k/24} \leq 2^{-12}\cdot 36 .$$

Hence, we get

$$\prod_{\substack{k\geq 1/\gamma:\\ s_k>0}} \left(\frac{\mathrm{e}\cdot 2^{\gamma\cdot k}\cdot c}{s_k\cdot 2^{\phi\cdot k/3}}\right)^{s_k} \leq \mathrm{e}^{\frac{\mathrm{e}\,c}{2}\sum_{k\geq 1/\gamma} 2^{-\phi\cdot k/4}} \leq 2^{\,c/48} .$$

Thus, altogether we obtain

$$\mathrm{E}[S_\lambda^{\mathcal{E}}] \leq 2^{-2\phi\lambda/3} \sum_{\langle s_1,\ldots,s_\lambda\rangle \in \mathcal{P}(\lambda;1/\gamma)} 2^{\,c/48} \leq 2^{-2\phi\lambda/3}\cdot|\mathcal{P}(\lambda;1/\gamma)|\cdot 2^{\,c/48} . \tag{8.1}$$

Obviously, for all $\lambda, \kappa \in \mathbb{N}$ it holds that

$$|\mathcal{P}(\lambda;\kappa)| \leq \begin{cases} 0 & : \quad \lambda < \kappa \\ 1 + \sum_{\lambda'=\kappa}^{\lambda-\kappa} |\mathcal{P}(\lambda';\kappa)| & : \quad \lambda \geq \kappa \end{cases} .$$

With this formula we can show the following claim.

Claim 8.20 *For all integers $\lambda, \kappa \geq 234$ it holds that $|\mathcal{P}(\lambda;\kappa)| \leq 2^{\lambda/36}$.*

Proof. We will prove the claim by induction on λ. Clearly, according to the formula for $|\mathcal{P}(\lambda;\kappa)|$ above, for all $\lambda \leq \kappa$ we have $|\mathcal{P}(\lambda;\kappa)| \leq 2^{\lambda/36}$. Now suppose that for some $\lambda > \kappa$ it has already been shown for all $\lambda' < \lambda$ that $|\mathcal{P}(\lambda';\kappa)| \leq 2^{\lambda'/36}$. Then it holds

$$\begin{aligned}
|\mathcal{P}(\lambda;\kappa)| &\leq 1 + \sum_{\lambda'=\kappa}^{\lambda-\kappa} |\mathcal{P}(\lambda';\kappa)| \\
&\leq 1 + \sum_{\lambda'=\kappa}^{\lambda-\kappa} 2^{\lambda'/36} \\
&\leq 1 + 36\cdot 2^{1+(\lambda-\kappa)/36} = 1 + 36\cdot 2^{1-\kappa/36}\cdot 2^{\lambda/36} \\
&\leq 1 + 0.8\cdot 2^{\lambda/36} \leq 2^{\lambda/36} .
\end{aligned}$$

□

Thus, for our choice of γ and ϕ we have $|\mathcal{P}(\lambda;1/\gamma)| \leq 2^{\phi\lambda/6}$. Using this in inequality (8.1) yields the lemma. □

8.3.2 Counting Valid 1-Neighborhood Sets

In this section we show how to apply Lemma 8.19 to bound the expected number of edge sets of certain size that form the 1-neighborhood of a valid 1,2-core witness. For this we first need some simple claims.

Claim 8.21 *For every edge $e \in E$ it holds*

$$\sum_{e' \in N(e)} 2^{-\delta \cdot |e'|} \leq \epsilon \cdot \ln 2 \cdot |e| \; .$$

Proof. Observe that from the assumption of Theorem 8.4 we obtain for every $e \in E$ that

$$\left(2^{1-|e|}\right)^{\epsilon} \leq 2^{-\delta \cdot |e|} \cdot \prod_{e' \in N(e)} \left(1 - 2^{-\delta \cdot |e'|}\right) \; .$$

This clearly implies that

$$2^{-\epsilon \cdot |e|} \leq \prod_{e' \in N(e)} \left(1 - 2^{-\delta \cdot |e'|}\right) \leq \prod_{e' \in N(e)} \exp\left(-2^{-\delta \cdot |e'|}\right) = \exp\left(- \sum_{e' \in N(e)} 2^{-\delta \cdot |e'|}\right) \; ,$$

which immediately yields the claim. □

Claim 8.22 *For every edge $e \in E$ and $k > 0$ it holds that*

$$|\{e' \in N(e) : |e'| \leq k\}| \leq 2^{\delta \cdot k} \cdot \epsilon \cdot \ln 2 \cdot |e| \; .$$

Proof. Follows directly from Claim 8.21 above. □

Claim 8.23 *Consider any $e \in E$. If we fix arbitrarily the colors of up to $\epsilon \cdot |e|$ nodes of e and then randomly color the other nodes, then it holds that*

$$\Pr[e \textit{ is bad}] \leq 2^{-|e|/2} \; .$$

Proof. Suppose that the colors of $s \leq \epsilon \cdot |e|$ nodes of e are determined. Then, clearly, the probability that e is bad is at most the probability that among the remaining $|e| - s$ nodes there are less than $2 \cdot \epsilon \cdot |e|$ nodes of one color. Therefore

$$\begin{aligned} \Pr[e \text{ is bad}] &\leq 2 \cdot \sum_{k=0}^{\lfloor 2\,\epsilon\,|e| \rfloor} \binom{|e| - s}{k} 2^{-(|e|-s)} \leq 4 \cdot 2^{-(|e|-s)} \cdot \binom{|e| - s}{2\,\epsilon\,|e|} \\ &\leq 4 \cdot 2^{-(|e|-s)} \cdot \left(\frac{\mathrm{e}}{2\,\epsilon}\right)^{2\,\epsilon\,|e|} \leq 4 \cdot 2^{-|e|(1-\epsilon)} \cdot \left(\frac{\mathrm{e}}{2\,\epsilon}\right)^{2\,\epsilon\,|e|} \; . \end{aligned}$$

For $\epsilon \leq 1/24$ this is at most $2^{-|e|/2}$ for any e with $|e| \geq 12/\epsilon^2$. □

Now we are ready to apply Lemma 8.19. Consider any core witness $\mathcal{E}$. Let the contribution Λ_e of an edge e be defined either as the size of e if e is bad, or 0 otherwise. Clearly, requirement (1) of Lemma 8.19 is fulfilled, since any edge in E must have a size of at least $1/\delta \geq 288$. Our assumption on $\mathcal{E}$ and Claim 8.22 together imply that

$$|\{e' \in N(\mathcal{E}) : |e'| \leq k\}| \leq 2^{\delta \cdot k} \cdot \epsilon \cdot \ln 2 \cdot \frac{|V_{\mathcal{E}}|}{1 - \epsilon} \leq \epsilon |V_{\mathcal{E}}| \cdot 2^{\delta \cdot k} \; .$$

Thus, requirement (2) holds with $c = \epsilon |V_{\mathcal{E}}|$. Requirement (3) follows from Claim 8.23 and the definition of a core witness.

Since all requirements of Lemma 8.19 are fulfilled, the expected number of possibilities $C^{(1)}_{\mathcal{E},\lambda}$ of choosing edges of contribution λ for a valid 1,2-core witness of $\mathcal{E}$ that belong to $N(\mathcal{E})$ satisfies

$$C^{(1)}_{\mathcal{E},\lambda} \leq 2^{-\lambda/4} \cdot 2^{\epsilon |V_{\mathcal{E}}|/48} \; . \tag{8.2}$$

8.3.3 Counting Valid 2-Neighborhood Sets

In this section we show how to apply Lemma 8.19 to bound the expected number of edge sets of certain size that form the 2-neighborhood of a valid 1,2-core witness.

Let us fix some core witness $\mathcal{E}$. We say an edge $e \in N(\mathcal{E})$ is of *type k* if there is a valid core witness $\mathcal{F} \in N(e) \setminus \mathcal{E}$ with $\sum_{e' \in \mathcal{F}} |e'| = k$. (Note that e can be of different types at the same time if there are different $\mathcal{F}$ of different size.) Recall from Definition 8.12 that in order e to be of type $k > 0$ the number of nodes in e covered by $\mathcal{F}$ must be at least $\epsilon \cdot |e|$. Therefore, we must have $|e| \leq k/\epsilon$. With this we can show the following claims.

Claim 8.24 *For every edge $e \in E$ and positive k it holds*

$$|\{e' \in N(e) : e' \textit{ can be of type in } \{1, \ldots, k\}| \leq 2^{k \cdot \delta/\epsilon} \cdot \epsilon \cdot \ln 2 \cdot |e| \ .$$

Proof. The proof of the claim follows directly from Claim 8.22. □

Claim 8.25 *For every $e \in E$ it holds that*

$$\Pr[e \textit{ is of type } k] \leq 2^{-k/6} \ .$$

Proof. If an edge e is of type k there must be a core witness $\mathcal{F} \in N(e)$ with $\sum_{e' \in \mathcal{F}} |e'| = k$. Since the size of e can be at most k/ϵ in order to be of type k, we get from formula (1) in Section 8.3.2 that

$$\Pr[e \text{ is of type } k] \ \leq \ 2^{-k/4} \cdot 2^{\epsilon |e|/48} \leq 2^{-k/6} \ .$$

□

For every edge $e_i \in N(\mathcal{E})$ let us introduce a set $\{e_{i,\epsilon |e_i|}, \ldots, e_{i,\lambda}\}$ of copies of e_i (λ will be specified below). We define the contribution $\Lambda_{e_{i,j}}$ of $e_{i,j}$ to be j if e_i is of type j and 0 otherwise. Now we apply Lemma 8.19 to the copies of all edges in $N(\mathcal{E})$. Clearly, requirement (1) of Lemma 8.19 holds, since every edge must have a type of either 0 or at least ϵ/δ. Requirement (2) follows from Claim 8.24 and requirement (3) follows from Claim 8.25 and the definition of a core witness. Thus, it follows from Lemma 8.19 that the expected number of possibilities $C^{(2)}_{\mathcal{E},\lambda}$ of choosing edges of contribution λ for a valid 1,2-core witness of $\mathcal{E}$ that belong to $N_2(\mathcal{E})$ satisfies

$$C^{(2)}_{\mathcal{E},\lambda} \leq 2^{-\lambda/12} \cdot 2^{\epsilon |V_{\mathcal{E}}|/48} \ . \tag{8.3}$$

8.3.4 Proof of Proposition 3.8

We are now ready to prove Proposition 8.17. Consider any fixed 1,2-core witness $\mathcal{E}$. Let the contribution of $\mathcal{E}$ be the sum of the edge sizes of its core edges. If $\mathcal{E}$ has contribution λ, then it must cover ν nodes, where $(1-\epsilon)\lambda \leq \nu \leq \lambda$ by Remark 8.7. Thus, we can apply the formulas (8.2) and (8.3) from Sections 8.3.2 and 8.3.3 to obtain the following bound for all $\nu \geq 6\epsilon\, |V_{\mathcal{E}}|$.

$$\begin{aligned}
\Pr[\hat{\nu}_{\mathcal{E}} \geq \nu] \ &\leq \ \mathrm{E}\left[|\{\mathcal{F} \ : \ \mathcal{F} \text{ is a 1,2-core witness of } \mathcal{E} \text{ and } |V_{\mathcal{F}}| \geq \nu\}|\right] \\
&\leq \ \sum_{\lambda \geq (1-\epsilon)\nu} \sum_{\kappa=0}^{\lambda} C^{(1)}_{\mathcal{E},\kappa} \cdot C^{(2)}_{\mathcal{E},\lambda-\kappa}
\end{aligned}$$

$$\begin{aligned}
&\leq \sum_{\lambda \geq (1-\epsilon)\nu} \sum_{\kappa=0}^{\lambda} 2^{-\kappa/4} \cdot 2^{\epsilon|V_{\mathcal{E}}|/48} \cdot 2^{-(\lambda-\kappa)/12} \cdot 2^{\epsilon|V_{\mathcal{E}}|/48} \\
&\leq 2^{\epsilon|V_{\mathcal{E}}|/24} \sum_{\lambda \geq (1-\epsilon)\nu} 2^{-\lambda/12} \sum_{\kappa=0}^{\lambda} 2^{-\kappa/6} \\
&= 2^{\epsilon|V_{\mathcal{E}}|/24} \sum_{\lambda \geq (1-\epsilon)\nu} 2^{-\lambda/12} \cdot \frac{1}{1-2^{-1/6}} \\
&= 2^{\epsilon|V_{\mathcal{E}}|/24} \cdot 2^{-(1-\epsilon)\nu/12} \cdot \frac{1}{(1-2^{-1/12}) \cdot (1-2^{-1/6})} \\
&\leq 164 \cdot 2^{-(1-\epsilon)\nu/12} \cdot 2^{\epsilon|V_{\mathcal{E}}|/24} \leq 164 \cdot 2^{-(1-\epsilon)\nu/12} \cdot 2^{\nu/144} \leq 2^{-\nu/24} .
\end{aligned}$$

From this it follows that

$$\mathrm{E}[\hat{\nu}_{\mathcal{E}}] \leq 6\epsilon \cdot |V_{\mathcal{E}}| + \sum_{\nu > 6\epsilon \cdot |V_{\mathcal{E}}|} 2^{-\nu/24} \leq |V_{\mathcal{E}}|/2 .$$

This completes the proof of Proposition 8.17. □

8.4 A randomized linear-time algorithm for 2-coloring

In this section we show how to modify the algorithm presented in Section 8.2 to obtain a randomized algorithm that returns a 2-coloring of a hypergraph in expected linear time. In the following, let m denote the number of edges and let M denote the sum of the sizes of all edges in the input hypergraph. Our algorithm runs in two phases.

Phase 1

In the first phase we run Step 1 and Step 2 of the algorithm presented in Section 8.2 (however, the ϵ there has to be replaced by $\sqrt{\epsilon}$).

Clearly, the time required for Step 1 is $O(M)$. Step 2 requires a careful implementation to obtain a runtime of $O(M)$. This can be achieved for Step 2.1 by ensuring the properties that

- for every hyperedge it has to be checked at most once via its nodes whether it is bad (afterwards, this property can be retrieved from some variable assigned to that hyperedge), and
- for every node covered by a 1-component, the set of edges adjacent to it is evaluated at most once (the hyperedges can store the number of nodes they have in common with a 1-component by some counter; this counter suffices to check whether an ϵ-fraction of a hyperedge has already been covered by the current 1-component).

In Step 2.2 it is easy to see that every hyperedge has to be considered at most once. Whatever decision is done for the hyperedge, it is removed afterwards from the set of edges that still have to be checked. Hence, the overall runtime of Phase 1 is $O(M)$.

In Section 8.3 we investigated the structure of the 2-components obtained. It follows from Theorem 8.10 that with high probability there is no 2-component of size larger than $c \log m$ for a sufficiently large c. As we will show, it follows also from the analysis presented in Section 8.3 that the expected number of 2-components of size k is at most $\frac{m}{2^{\xi k}}$ for a suitably chosen constant

ξ. Indeed, since each 2-component of size k has to start from a bad edge e of size less than or equal to k, the expected number of 2-components of size k is bounded by the following sum:

$$\sum_{e:\,|e|\le k/(2\,c)} \Pr[\text{2-component } \mathcal{W} \text{ having } V_{\mathcal{W},0} = \{e\} \text{ is of size } k] \;+ \sum_{e:\,|e|\in\{k/(2\,c)+1,\dots,k\}} \Pr[e \text{ is bad}]\;,$$

where c is chosen as in Proposition 8.18. We can apply Propositions 8.17 and 8.18 to bound the first term from above by $|\{e \;:\; |e| \le k/(2\,c)\}| \cdot 2^{-k/(2\,c)}$. By Claim 8.23, the second term is bounded by $|\{e \;:\; |e| > k/(2\,c)\}| \cdot 2^{-k/(4\,c)}$. Hence, the expected number of 2-components of size k is at most

$$\sum_{e:\,|e|\le k/(2\,c)} 2^{-k/(2\,c)} + \sum_{e:\,|e|>k/(2\,c)} 2^{-k/(4\,c)} \le m \cdot 2^{-k/(4\,c)}\;.$$

Phase 2

In the second phase we consider each 2-component $\mathcal{C}$ obtained after Phase 1 independently. For every $\mathcal{C}$ we run Steps 1 and 2 (again, with ϵ replaced by $\sqrt{\epsilon}$). Let $\mathcal{E}_{\mathcal{C}}$ be the event that all of the 2-components obtained out of $\mathcal{C}$ after performing Steps 1 and 2 are of size at most $\varsigma \log |V_{\mathcal{C}}|$, where ς is a suitably chosen constant and $V_{\mathcal{C}}$ is the set of the nodes covered by the edges in $\mathcal{C}$. We repeat independently Steps 1 and 2 for $\mathcal{C}$ until the event $\mathcal{E}_{\mathcal{C}}$ holds. Then, we perform Step 3 as described in Section 8.2.

In the following we estimate the running time of Phase 2. Since we are using the same algorithm as for the whole hypergraph, we can apply our analysis presented in Section 8.3 to study the 2-components obtained in Phase 2 out of the 2-component $\mathcal{C}$. We start with the first part in which, for every 2-component $\mathcal{C}$ obtained after Phase 1, we repeat Steps 1 and 2 until the event $\mathcal{E}_{\mathcal{C}}$ holds. Then, by Claim 8.22, there are at most $|V_{\mathcal{C}}| \cdot 2^{\delta' k}$ edges of (possibly reduced) size at most k in $\mathcal{C}$, where $\delta' = \delta/\sqrt{\epsilon}$. Since $\Pr[e \text{ is bad}] \le 2^{-|e|/2}$, the probability that there is a bad edge e in $\mathcal{C}$ that has a size of larger than or equal to $c_1 \log |V_{\mathcal{C}}|$ is bounded from above by

$$\begin{aligned}\sum_{\substack{e\in\mathcal{C}:\\ |e|\ge c_1 \log|V_{\mathcal{C}}|}} 2^{-|e|/2} &\le \sum_{k\ge c_1 \log|V_{\mathcal{C}}|} |V_{\mathcal{C}}| \cdot 2^{\delta' k} \cdot 2^{-k/2} \le \sum_{k\ge c_1 \log|V_{\mathcal{C}}|} |V_{\mathcal{C}}| \cdot 2^{-k/3} \\ &\le 5 \cdot |V_{\mathcal{C}}| \cdot 2^{-c_1 \log|V_{\mathcal{C}}|/3}\;.\end{aligned}$$

Therefore, if we choose c_1 sufficiently large, then with probability at least $1 - 1/|V_{\mathcal{C}}|$ each $V_{\mathcal{C},0}$ obtained in Step 2 is of size not larger than $c_1 \log |V_{\mathcal{C}}|$. Similar arguments (by applying Propositions 8.17 and 8.18) imply that with probability at least $1 - 1/|V_{\mathcal{C}}|$ each 2-component within $\mathcal{C}$ will cover at most $\varsigma \log |V_{\mathcal{C}}|$ nodes, for a sufficiently large constant ς. Therefore, the expected running time in Steps 1 and 2 until the event $\mathcal{E}_{\mathcal{C}}$ holds is of the order of the sum of the sizes of the (possibly reduced) edges in $\mathcal{C}$. Summing this over all 2-components, the expected running time is at most $O(M)$.

Once the event $\mathcal{E}_{\mathcal{C}}$ holds, the running time of Step 3 for all the 2-components $\mathcal{C}_1, \dots, \mathcal{C}_r$ obtained from $\mathcal{C}$ in Phase 2 is bounded by

$$\begin{aligned}\sum_{i=1}^{r} O\left(2^{|V_{\mathcal{C}_i}|} \cdot \sum_{e\in\mathcal{C}_i} |e|\right) &= \sum_{i=1}^{r} O\left(2^{|V_{\mathcal{C}_i}|} \cdot \left(2^{|V_{\mathcal{C}_i}|} \cdot |V_{\mathcal{C}_i}|\right)\right) \\ &= O\left(r \cdot \left(2^{\varsigma \log|V_{\mathcal{C}}|}\right)^2 \cdot |V_{\mathcal{C}}|\right) = O(|V_{\mathcal{C}}|^{2\varsigma+2})\;,\end{aligned}$$

where $V_{\mathcal{C}_i}$ is the set of nodes in $\mathcal{C}_i$ at the beginning of Step 3 in Phase 2 and we have used the trivial upper bound $|\mathcal{C}_i| \leq 2^{|V_{\mathcal{C}_i}|}$.

Since after Phase 1 the expected number of 2-components with k nodes is bounded by $\frac{m}{2^{\xi k}}$, the expected time required to perform Step 3 for all 2-components is bounded by

$$\sum_{\text{2-components } \mathcal{C} \text{ of Phase 1}} O(|V_{\mathcal{C}}|^{2\varsigma+2}) = O\left(\sum_{k=1}^{c \log M} k^{2\varsigma+2} \cdot \frac{m}{2^{\xi k}}\right) = O(m) \, .$$

Hence, the total expected running time of our algorithm is $O(M)$.

8.5 Extensions to c-coloring

In this section we extend our results to the c-coloring problem. First, we observe that the algorithms presented in the previous sections can be easily extended to deal with any c-coloring. A simple modification of the proof of Theorem 8.4 yields the following theorem.

Theorem 8.26 *Let $c \geq 2$ be an arbitrary constant. Then there exist constants K, Δ, $\mathcal{E} > 0$ such that for any $k \geq K$, $0 < \delta \leq \Delta$, and $0 < \epsilon \leq \mathcal{E}$ it holds:*

Consider any hypergraph H with edges $e_1, \ldots, e_m$, in which every edge is of size at least k. Let A_i be the event that e_i is monochromatic, $1 \leq i \leq m$. Further, let $x_i = c^{-\delta\,|e_i|}$ for all i, $1 \leq i \leq m$. If it holds that

$$(\Pr[A_i])^{\epsilon} \leq x_i \prod_{e_j \in N(e_i)} (1 - x_j)$$

for all i, $1 \leq i \leq m$, in the case that the color of each node is chosen i.u.r., then there is a randomized algorithm that finds a c-coloring of H in expected time linear in $\sum_i |e_i|$.

One of the restrictions in Theorems 8.4 and 8.26 is that the size of the hyperedges must exceed some constant. In the following theorem we demonstrate that this condition can already be avoided if 4 colors are available.

Theorem 8.27 *There exist constants $\Delta', \mathcal{E}' > 0$ so that for any $0 < \delta \leq \Delta'$ and $0 < \epsilon \leq \mathcal{E}'$ it holds:*

Consider any hypergraph H with edges $e_1, \ldots, e_m$ of any size. Let A_i be the event that e_i is monochromatic. Further, let $x_i = 4^{-\delta\,|e_i|}$. If it holds that

$$(\Pr[A_i])^{\epsilon} \leq x_i \prod_{e_j \in N(e_i)} (1 - x_j) \tag{8.4}$$

for all $i \in \{1, \ldots, m\}$, in the case that the color of each node is chosen i.u.r., then there is a randomized algorithm that finds a 4-coloring of H in polynomial time.

Proof. Let us choose $\mathcal{E}' = \mathcal{E}/6$ and $\Delta' = \Delta/2$, where $\mathcal{E} = 1/24$ and $\Delta = \mathcal{E}^2/12$. From Theorem 8.4 and Remark 1.6 we know that in this case there is a 2-coloring for all hyperedges in H of size at least $1/\Delta$. Thus, there are 2 colors left we can use to color the hyperedges of size below $1/\Delta$. All possible combinations of the 2 sets of 2 colors require only 4 colors, and hence a 4-coloring of H could be found if the small hyperedges could be 2-colored.

For any edge e_j of the small edges it clearly holds that $x_j = 4^{-\delta\,|e_j|} \geq 2^{-\Delta\,|e_j|} \geq 1/2$. Hence, $1 - x_j \leq 1/2$ and therefore $\Pr[A_i]^{\epsilon} \leq x_i \prod_{e_j \in N(e_i)} (1 - x_j)$ for $|N(e_i)| > 0$ (w.r.t. small edges) only if $4^{-\epsilon|e_i|} \leq 1/2$. This is true if and only if $|e_i| \geq 1/(2\epsilon)$. Since $\epsilon \leq \mathcal{E}'$, any e_i with $|N(e_i)| > 0$ has to have a size of at least $3/\mathcal{E}$. Hence, all e_i with $|e_i| < 3/\mathcal{E}$ are isolated from all other small edges and thus can be colored without any problem. It therefore remains to consider only hyperedges of size $k \in \{3/\mathcal{E}, \ldots, 1/\Delta\}$. For each of these edges the probability to become monochromatic is at most $2 \cdot 2^{-k}$, and any of these edges can intersect at most $(\mathcal{E}/3) \cdot k$ other edges without violating inequality (8.4). Set $p = 2^{-3/\mathcal{E}+1}$ and $d = (\mathcal{E}/3) \cdot (1/\Delta)$. Since $p \cdot d^9 \leq (1/2\mathrm{e})^3$ (which is easy to check), we can apply the algorithm by Molloy and Reed (see Section 3.6.2) to find a 2-coloring also for the small hyperedges. □

Theorem 8.27 allows to prove the following result.

Theorem 8.28 *There exist constants $\Delta', \mathcal{E}' > 0$ so that for any $0 < \delta \leq \Delta'$ and $0 < \epsilon \leq \mathcal{E}'$ it holds:*

Consider any hypergraph H with n nodes and m edges $e_1, \ldots, e_m$ of any size. Let $c \geq 4$ be an arbitrary integer that may depend on n and m. Furthermore, A_i be the event that e_i is monochromatic, and let $x_i = c^{-\delta|e_i|}$. If it holds that

$$(\Pr[A_i])^\epsilon \leq x_i \prod_{e_j \in N(e_i)} (1 - x_j) \tag{8.5}$$

for all $i \in \{1, \ldots, m\}$, in the case that the color of each node is chosen i.u.r., then there is a randomized algorithm that finds a c-coloring of H in polynomial time.

Proof. Theorem 8.27 implies that if Δ' and $\mathcal{E}'$ are chosen sufficiently small, then the problem of coloring the hyperedges decomposes into three independent problems:

(1) small, isolated hyperedges,

(2) hyperedges that can be colored using Molloy and Reed's method, and

(3) hyperedges that can be colored using the algorithm for Theorem 8.4.

Case (1) hyperedges cause no problem. Case (2) hyperedges can be colored in polynomial time for any c with the help of the improved form of Molloy and Reed's method given at the end of Section 3.6.2.

For the case (3), observe that according to Section 8.4 the hyperedges can be decomposed into components of size at most $O(\log\log m)$. If $c = (\log m)^{O(1)}$, then one can find via exhaustive search in polynomial time a coloring such that no hyperedge is monochromatic. If $c = (\log m)^{\omega(1)}$, then we perform another random experiment of coloring the nodes in each component. Since a component of size $O(\log\log m)$ can have at most $(\log m)^{O(1)}$ hyperedges and the probability of a hyperedge to become monochromatic is at most $1/c$, it follows that with high probability all hyperedges in the component will be non-monochromatic. Thus, also in case (3) we end up with a polynomial-time algorithm. □

8.6 Proof of Proposition 8.18

To show the first claim of Proposition 8.18, let us observe that since $\mathrm{E}[X_{i+1} \mid X_0, X_1, \ldots, X_i] \leq \beta \cdot X_i$, we immediately obtain $\mathrm{E}[X_t] \leq \beta^t \cdot \lambda$. This implies that

$$\mathrm{E}[\sum_{j \geq 0} X_j] \leq \sum_{j \geq 0} \beta^j \cdot \lambda = \tfrac{1}{1-\beta} \cdot \lambda \leq 2 \cdot \lambda$$

since $\beta \leq 1/2$. Now we prove the second claim. Since $\mathrm{E}[X_t] \leq \beta^t \cdot \lambda$, it follows from the Markov Inequality that

$$\Pr[X_\tau > 0] \leq \beta^\tau \cdot \lambda \leq \frac{1}{2\,s} \tag{8.6}$$

for $\tau \geq \log_{1/\beta}(2\,s\,\lambda)$. Let us now fix $\tau = \max[\log_{1/\beta}(2\,s\,\lambda), \ln(2s)]$ and condition on the event $X_\tau = 0$.

For a given sequence $X_0, \ldots, X_\tau$, we say X_i is *ascending* if $i \geq 1$ and $X_i > \frac{1}{2} \cdot X_{i-1}$. Observe that if $X_j, X_{j+1}, \ldots, X_s > 0$ are not ascending, then they decrease at least geometrically. Therefore, if $X_{i_1}, X_{i_2}, \ldots, X_{i_r}$ are all ascending random variables in $X_0, \ldots, X_\tau$, then we have

$$\sum_{j \geq 0} X_j \leq 2 \cdot \lambda + \sum_{j=1}^{r} 2 \cdot X_{i_j} \ . \tag{8.7}$$

Thus from now on our aim is to bound $\sum_{j=1}^r X_{i_j}$.

Let $Y_1, \ldots, Y_\tau$ be independent, geometrically distributed random variables with $\Pr[Y_j = t] = \alpha^{t-1}(1-\alpha)$ for every $t \in \mathbb{N}$. It is easy to see that $\Pr[Y_j \geq t] = \alpha^{t-1}$ for all $t \in \mathbb{N}$. Since, by assumption, $\Pr[X_j \geq t] \leq \alpha^t$ for all $t \geq \frac{1}{2} X_{j-1}$, $(Y_j - 1)$ stochastically dominates any ascending X_j. Thus, it follows from Lemma 3.41 and (8.7) that $\sum_{j\geq 0} X_j$ is stochastically dominated by $2 \cdot (\lambda + \sum_{j=1}^\tau (Y_j - 1))$. Hence, it remains to bound $\sum_{j=1}^\tau Y_j$. From Section 3.2.9 we know that $\mathrm{E}[Y_j] = 1/(1-\alpha)$ for all j and therefore $\mathrm{E}[Y] = \tau/(1-\alpha)$. Thus, it follows from Theorem 3.38 that for any $\delta \geq 3$,

$$\begin{aligned}\Pr[Y \geq (1+\delta)\mathrm{E}[Y]] &\leq (1+\delta)^\tau \mathrm{e}^{-\delta \cdot \tau} = \mathrm{e}^{(\ln(1+\delta)-\delta)\tau} \\ &\leq \mathrm{e}^{-\tau} \ .\end{aligned}$$

Thus,

$$\Pr[Y \geq 4\tau/(1-\alpha)] \leq \mathrm{e}^{-\tau} \leq \frac{1}{2\,s}$$

for τ chosen as above. Therefore, we have

$$\Pr\left[\sum_{j\geq 0} X_j \geq 2(\lambda + (4/(1-\alpha) - 1)\tau) \mid X_\tau = 0\right] \leq \frac{1}{2\,s} \ .$$

Together with (8.6) this implies

$$\Pr\left[\sum_{j\geq 0} X_j \geq 2\left(\lambda + (\tfrac{4}{1-\alpha} - 1)\tau\right)\right] \leq \frac{1}{s} \ .$$

8.7 Conclusions

We presented methods in this chapter that allow us to find 2-colorings for hypergraphs in polynomial time, and in some cases even in expected linear time, for which no polynomial time algorithm was previously known.

There are still many open problems left. For instance, are there other techniques than the techniques based on the Lovász Local Lemma that allow to find 2-colorings for a much larger class of hypergraphs? Where are the limits of our approach? For instance, what is the largest ϵ for which our algorithm still runs in polynomial time? $\epsilon = 1/6$ might already work to obtain a polynomial-time algorithm for the 2-coloring problem. Is there a polynomial-time algorithm that finds a 2-coloring if there is no restriction on the minimum size of the hyperedges?

Chapter 9

A General Algorithmic Approach to the LLL

The Lovász Local Lemma (LLL) is a powerful tool that is increasingly playing a valuable role in computer science. It has led to solutions for numerous problems in many different areas, reaching from problems in pure combinatorics to problems in routing, scheduling and approximation theory. However, since the original lemma is non-constructive, many of these solutions were first purely existential. A breakthrough result by Beck [Bec91] and its generalizations (Alon [Alo91], Leighton, Maggs and Richa [LMR99], Molloy and Reed [MR98]) have led to polynomial time algorithms for many of these problems. However, these methods can only be applied to a simple, symmetric form of the LLL. In this chapter, we provide an general approach to design polynomial-time algorithms for problems that require the LLL in its general form. It extends the techniques presented in the previous chapter to find good approximate solutions to a large class of NP-hard problems called *minimax integer programs* (MIPs). As we will demonstrate, this method finds approximate solutions that are — especially for problems of non-uniform character — significantly better than all methods presented before. In particular, we will show that our method allows us to transform the existential job shop scheduling results presented in Chapter 7 into algorithms that find schedules close to the predicted solutions in polynomial time. Furthermore, we demonstrate how our results can be used to solve satisfiability problems.

9.1 Introduction

The probabilistic method is used to prove the existence of objects with desirable properties by showing that a randomly chosen object from an appropriate probability distribution has the desired properties with positive probability. In most applications, this probability is not only positive but is actually high and frequently tends to 1 as the parameters of the problem tend to infinity. In such cases, the proof usually supplies an efficient randomized algorithm for producing a structure of the desired type.

There are, however, certain examples, where one can prove the existence of the required combinatorial structure by probabilistic arguments that deal with rare events; events that hold with positive probability which is exponentially small in the size of the input. This happens often when using the LLL.

As noted in the previous chapters, a number of algorithms have already been found for applications of the symmetric form of the Lovász Local Lemma, but almost no algorithm has been found so far for applications that require the Lovász Local Lemma in its general form. In this chapter, we develop techniques that allow us to exploit non-uniform dependences in order to find good approximation ratios for so-called *minimax integer programs* (MIPs).

Definition 9.1 *A MIP has variables* $\{x_{i,j} \; : \; i \in [n], \; j \in [\ell_i]\}$*, for some integers* ℓ_i*. Let* $N = \sum_{i\in[n]} \ell_i$ *and let* $\boldsymbol{x}$ *denote the* N*-dimensional vector of the variables* $x_{i,j}$*. A MIP seeks to minimize a real* Y *subject to:*

(1) a system of linear inequalities $\boldsymbol{A}\boldsymbol{x} \le \boldsymbol{y}$*, where* $\boldsymbol{A} \in [0,1]^{m\times N}$ *and* $\boldsymbol{y}$ *is the* m*-dimensional vector with the variable* Y *in each component,*
(2) for all $i \in [n]$*:* $\sum_{j\in[\ell_i]} x_{i,j} = 1$*, and*
(3) for all i *and* j*:* $x_{i,j} \in \{0,1\}$*.*

MIPs represent a general framework for choice problems (see properties (2) and (3)). Many applications in the area of routing and scheduling can be formulated as MIPs (for several examples, see [LRS99]). In general, MIPs are NP-hard. A common strategy to find good approximate solutions to MIPs is to first solve its linear relaxation and then to use randomized rounding. This approach was pioneered by Raghavan and Thompson [RT87] and was later extended and refined for MIPs by Srinivasan [Sri96], Lu [Lu99] and Leighton, Rao, and Srinivasan [LRS99]. The latter papers apply the symmetric form of the LLL to construct good rounding strategies. Leighton et al. [LRS99] give as a high-level application of MIPs the hypergraph partitioning problem (see Section 3.6.1):

Let $H = (V, E)$ be a set system with $V = [n]$ and $E = \{S_1, \dots, S_M\} \subseteq \mathcal{P}(V)$. Given a positive integer ℓ, the problem is to partition V into ℓ parts, so that the sets are split well: we want a $\chi : V \to [\ell]$ that minimizes $Y = \max_{j\in[M],k\in[\ell]} |\{i \in S_j : \; \chi(i) = k\}|$.

The cost function above has the drawback that it only seeks to minimize the *absolute* split size. However, there are applications where it would be much more desirable to minimize the *relative* split size, i.e., to find a function χ that minimizes $Y = \max_{j\in[M],k\in[\ell]} |\{i \in S_j : \; \chi(i) = k\}|/|S_j|$. Problems of this kind can be found, for instance, in Chapter 7. Their solution (which heavily uses the general LLL) allowed them to prove upper bounds on the makespan of job shop schedules that significantly improved the previously best results. Solving these problems with the techniques given in [LRS99, Lu99] gives an approximation ratio that is as large as $\Theta(\frac{\log n}{\log \log n})$ (n is the problem size), whereas an approximation ratio of partly as low as $1 + o(1)$ is required for the proofs in Chapter 7 to be constructivized. Our method presented in the following will achieve this goal.

9.1.1 New Results

In this chapter, we present an approach to design polynomial-time algorithms for problems that require the general LLL. We will apply this approach to find good approximation algorithms for MIPs:

Consider any MIP. Our strategy is to start with an optimal solution $\{x^*_{i,j} \; : \; i \in [n], \; j \in [\ell_i]\}$ to the LP relaxation of the MIP (i.e., the integrality constraints in Definition 9.1 (3) are removed). The resulting LP optimum $\boldsymbol{y}^*$ is clearly a lower bound for $\boldsymbol{y}$, the optimum of the (integral form of the) MIP. We will present a randomized rounding algorithm that exploits the dependencies among the rows of matrix $\boldsymbol{A}$ to find a good approximate solution for $\boldsymbol{y}$. These dependencies are defined as follows.

Definition 9.2 *Given a MIP instance I as defined above, let the* dependency graph G_I *of I consist of the node set $[m]$ and an edge set that contains edge (r,s) if and only if there are i,j,j' so that $a_{r,(i,j)} > 0$ and $a_{s,(i,j')} > 0$, and $r \neq s$ or $j \neq j'$.*

Note that G_I may contain self-loops. This simplifies our proofs. However, it would also be possible to avoid them. Based on our LLL techniques, we will prove the following main result.

Theorem 9.3 *Given a MIP instance I, let $\boldsymbol{x}^*$ be its optimal fractional solution and $\boldsymbol{y}^*$ be the vector with $y_r^* = (\boldsymbol{A}\boldsymbol{x}^*)_r$ for all r. Consider the random experiment of setting $x_{i,j}$ to 1 and all other $x_{i,j'}$'s to 0 with probability $x_{i,j}^*$. For any vector $\boldsymbol{\alpha} \in \mathrm{IR}_+^m$, let*

$$p_r = \mathrm{e}^{-\min[\alpha_r,\, \alpha_r^2] y'_r/3}$$

for all $r \in [m]$, where $y'_r = y_r^*/(\max_c a_{r,c})$. ($p_r$ is a Hoeffding bound for $\Pr[(\boldsymbol{A}\boldsymbol{x})_r \geq (1+\alpha_r)y_r^*]$.) Let $0 < \epsilon < 1$ be a constant that is chosen such that $\alpha_r \geq \max[(y'_r)^{-1},\ (y'_r)^{-(1-\epsilon)/2}]$ for every r. Furthermore, let $q_r = \mathrm{e}^{-\ln^\epsilon p_r^{-1}}$ and let $z_r = q_r^\delta$ be at most $1/3$, where δ is a sufficiently small positive constant. If it holds that

$$q_r \leq z_r \prod_{(r,s) \in G_I} (1 - z_s)^{2/\epsilon^2} \tag{9.1}$$

for all r, then there is an algorithm that finds a vector $\boldsymbol{x}$ in polynomial time so that $(\boldsymbol{A}\boldsymbol{x})_r \leq (1 + O(\alpha_r))y_r^*$ for all r.

The main feature of this result is its great flexibility: It enables us to exploit local dependencies among the rows to achieve approximate solutions that allow us to distinguish between rows that require a very good or just a rough approximation. This is important, since MIPs may be — especially in real-world applications — composed of different classes of restrictions that have different dependency structures and that require different approximation ratios. In [LRS99, Lu99] only the maximum dependency of a row on other rows can be used.

As we will see in Section 9.4, the property of being able to exploit local dependencies is vital to constructivize the job shop scheduling results presented in Chapter 7. We also show how to apply our method to obtain the first approximation algorithm for MAX SAT problems that is able to exploit local dependencies.

9.2 The Local Lemma Algorithm

We start with presenting a general approach to design polynomial-time algorithms for problems that require the general LLL.

Let V denote a set consisting of n independent random variables called *trials*. For each trial $t \in V$, let Ω_t denote the set of its possible outcomes. Let $\mathcal{A}$ be a set of m "bad" events, where each event $A \in \mathcal{A}$ is determined by the outcome of the trials in $T_A \subseteq V$. Let $E = \{T_A \mid A \in \mathcal{A}\}$. Two events $A_1, A_2 \in \mathcal{A}$ are said to be *neighbors* if $T_{A_1} \cap T_{A_2} \neq \emptyset$. For every $A \in \mathcal{A}$, let $N(A)$ denote the set of all neighbors of A. Clearly, event A is mutually independent of every subset of events A' with $A' \notin N(A)$. We assume that for every event A and every set $S \subseteq T_A$, the event A *restricted to* S (written as $A|_S$) is well-defined. Let $\Pr[A|_S]$ denote the probability that $A|_S$ is true under the assumption that an outcome is chosen independently at random for each trial in S. Suppose that the following natural assumptions are valid:

- For every trial, a random outcome can be generated in polynomial time, and
- it can be decided in polynomial time for every event A and set $S \subseteq T_A$ whether $A|_S$ is true.

Then the following theorem holds.

Theorem 9.4 *There is a constant $\Delta > 0$ so that for every $0 < \delta \leq \Delta$ the following holds:*

Let $A_1, \ldots, A_m$ be a set of "bad" events over a set V of independent random trials, $|V| = n$. Every trial $t \in V$ has a set of outcomes of size at most polylogarithmic in $n + m$. Let $p_i = \mathrm{e}^{-|T_{A_i}|^\epsilon}$ for all $i \in [m]$, where $0 < \epsilon \leq 1$ is a constant away from 0. Suppose that for all $S \subseteq T_{A_i}$ it holds that

- *if $|S| > |T_{A_i}|^\epsilon$ then $\Pr[A_i|_S] \leq p_i$, and*
- *if $|S| \leq |T_{A_i}|^\epsilon$ then $\Pr[A_i|_S] = 0$.*

Furthermore, assume that for $q_i = \mathrm{e}^{-\ln^\epsilon p_i^{-1}}$ and $x_i = q_i^\delta$ it holds that $x_i \leq 1/\mathrm{e}$ and

$$q_i \leq x_i \prod_{A_j \in N(A_i)} (1 - x_j)$$

for all i. Then there is a randomized algorithm that finds outcomes for the random trials in polynomial time (in $n + m$), w.h.p., such that for every event A, the set T_A can be partitioned into at most some constant number of subsets $S_1, \ldots, S_k$ so that $A|_{S_j}$ is false for all $j \in [k]$.

We first describe the algorithm underlying Theorem 9.4. Its analysis will be given in Section 9.2.2.

9.2.1 Description of the Algorithm

Consider any set of events that fulfills the conditions in Theorem 9.4. Our algorithm consists of four steps.

Step 1: Choose a random outcome for every trial in V.

Before we describe Step 2, we first introduce some notation and provide the ideas behind that step. Let the events in Theorem 9.4 be called *basic events*, and let any event defined by a basic event restricted to a subset of its trials be called a *reduced event*. As in Theorem 9.4, we assume the basic events to be numbered consecutively from A_1 to A_m. Given an event B representing an event A_i or any reduced event of A_i, we denote its *index* by i. A basic event A is called *bad* if its trial set cannot be decomposed into a constant number of subsets S such that $A|_S$ is false. Clearly, after Step 1 there might be many basic events left that are bad. In this case we have to redo certain trials covered by these events. The aim of Step 2.1 is to find a partition of the bad events into trial-disjoint groups. The key feature of our partitioning procedure is that we do not consider all the trials covered by the bad events. Instead, in the course of the algorithm we shall sometimes partition the basic events into several reduced events that are assigned to different groups.

Step 2: Perform the following two substeps Step 2.1 and Step 2.2.

Step 2.1:

set $R = V$ // R denotes the set of remaining trials
for $i = 1$ to m do:
  if $A_i|_{T_{A_i} \cap R}$ is true **then**
    call **Build_1-Component(**$A_i|_{T_{A_i} \cap R}$**)**

Algorithm Build_1-Component(B**):**

set $i = 0$ and $E_0 = \{B\}$
set $R = R \setminus T_B$
repeat
  set $E_{i+1} = \emptyset$
  set $C_{i+1} = N(E_i)$ // C_{i+1} = all basic events that are neighbors of E_i
  for all events $A_j \in C_{i+1}$ in increasing j do:
    set $S = T_{A_j} \cap R$ // S: trials not selected yet
    if $A_j|_S$ is true **then** // if true, this implies $|S| > |T_{A_j}|^\epsilon$
      set $R = R \setminus S$ and add $A_j|_S$ to E_{i+1}
  set $i = i + 1$
until $E_i = \emptyset$

Consider some fixed set $E^* = \bigcup_{i \geq 0} E_i$ of (possibly reduced) events built in the algorithm Build_1-Component. A basic event A is said to *participate* in it if more than $|T_A|^\epsilon$ of its trials are covered by E^*. (This clearly holds for those events that have a reduced event in E^*.) The set of all participating events restricted to the trials covered by E^* is called a *1-component*. For every 1-component C, let $B_C = E^*$ denote its set of *core events*. Core events play an important role in the reselection process because, as we shall show, it is sufficient to reselect only the trials covered by the core events in order to find a good selection for all (possibly reduced) events in C.

After performing Step 2.1, one might think that the 1-components can be solved independently. However, it could happen that some basic event is partitioned into too many reduced events that belong to different components. Theorem 9.4 requires for every event A to provide a decomposition of T_A into a constant number of sets S such that $A|_S$ is false. Therefore, without a coordination among these components, Theorem 9.4 might not hold at the end. Step 2.2 takes care of these events by combining components into one component if necessary. As we will see, the trials covered by the components constructed in Step 2.2 can afterwards be reselected independently so that now Theorem 9.4 is ensured to hold at the end.

Define $\mathcal{C}$ to be the set of all 1-components. Recall that $\mathcal{A}$ denotes the set of all basic events. The second substep works as follows.

Step 2.2:

set $S = \mathcal{A}$ // S denotes the set of remaining events
set $R = \mathcal{C}$ // R denotes the set of remaining 1-components
for all not yet taken 1-components C in R in
    increasing index of their initial events do:
  call **Build_2-Component(**C**)**

Algorithm Build_2-Component(C**):**

set $i = 0$ and $E_0 = B_C$
repeat
  set $E_{i+1} = \emptyset$

set $D = \bigcup_{A \in E_i}(N(A) \cap S)$ // $D =$ all neighboring basic events not yet considered
for all events $A_j \in D$ in increasing j do: // check nearby 1-components
  set T to the set of trials in A_j covered by outside 1-components
  if $|T| > |T_{A_j}|^\epsilon$ **then** $(*)$
    for all 1-components C' in R that overlap with T_{A_j} do:
      set $E_{i+1} = E_{i+1} \cup B_{C'}$
      remove C' from R
  remove A_j from S
set $i = i + 1$
until $E_i = \emptyset$

In the following, a basic event A is called *dangerous* if condition $(*)$ above holds for it. Let the union of the 1-components built in the algorithm Build_2-Component be called a *2-component*. For every 2-component C, let B_C denote the union of the sets $B_{C'}$ of the 1-components C' it consists of and let V_C be the set of trials covered by the events in B_C. Every basic event participating in 1-components in C is also said to *participate* in C, but only as a single event (by combining the reduced events from different 1-components into one event).

Remark 9.5 *Let us notice some important properties that we shall frequently use in our analysis and which follow immediately from our construction:*

(1) For every 2-component C, all events in B_C are true.
(2) For every 2-component C, the trial sets covered by the core events $A' \in B_C$ are disjoint and of size more than $|T_A|^\epsilon$ of the corresponding basic event A.
(3) Every basic event has at most one reduced event that is a core event.
(4) For every basic event A, the part of A not covered by 2-components is false.

For the next step we use the following properties.

Lemma 9.6

(1) For every 2-component C, there are outcomes for the trials in V_C so that all events participating in C are false.
(2) For every basic event A, A participates in at most one 2-component.
(3) For every basic event A, the part of A not participating in any 2-component cannot become bad if only trials covered by the 2-components are redone.

Proof. To prove (1), observe that for every 2-component C, every basic event participates as at most one (possibly reduced) event in C. Thus, one can easily verify that the probability bounds in Theorem 9.4 together with the LLL imply that there are outcomes for the trials in V_C so that all events participating in C are false.

For (2), assume that there is a basic event A that participates in at least 2 different 2-components. In this case, A must have been dangerous from the viewpoint of every 1-component. This, however, would have caused combining all trials of A covered by 1-components into a single 2-component, contradicting the assumption.

In order to prove (3), note that, by Remark 9.5 (4), the part of an event not covered by 2-components is false. Furthermore, the part of an event covered by 2-components in which it does not participate is small enough that it can be divided into at most two parts that cannot become true. Otherwise, $(*)$ of Step 2.2 would have been true. Hence, the part of an event not participating in any 2-component cannot become bad if only trials covered by the 2-components are redone. □

Remark 9.5 and Lemma 9.6 imply the following vital property.

Corollary 9.7 *The 2-components can be considered independently of each other in order to fulfill Theorem 9.4.*

Now we are ready to present Step 3.

Step 3: For every 2-component C, apply independently Step 2 above until the 2-components resulting from C are sufficiently small.

From Section 9.2.2 it follows that we can identify "sufficiently small" with 2-components covering at most $O(\ln^{1/\epsilon} \ln m)$ trials. Since by Theorem 9.4 every trial only has a set of outcomes that is polylogarithmic in $n + m$, we can use the following last step.

Step 4: Find through exhaustive search outcomes for the trials in each 2-component so that all of its events are false.

9.2.2 Analysis of the Algorithm

The following lemma implies that the overall runtime of the algorithm presented above is polynomial in $n + m$.

Lemma 9.8 *Let ϵ and δ be chosen as in Theorem 9.4. Then,*

(1) with probability at least $1 - 1/m^\alpha$ for any constant $\alpha > 0$ the number of trials covered by any 2-component after Step 2 is at most $O((\alpha \ln m)^{1/\epsilon})$, and

(2) for any 2-component C after Step 2 of size $O(\ln^{1/\epsilon} m)$, with probability at least $1 - 1/\ln^\alpha m$ for any constant $\alpha > 0$ the number of trials covered by any 2-component in C after applying Step 2 to C is at most $O(((\alpha + 1/\epsilon) \cdot \ln \ln m)^{1/\epsilon})$.

In order to prove the lemma, we intend to bound the expected number of possibilities of choosing sets of core events that are able to establish a 2-component. We will do this by counting all possible sets of basic events that could represent the core events of a 2-component. There are two main problems that have to be solved for the counting:

(1) First of all, the counting has to provide an ordering for these basic events. The ordering must be able to *uniquely* determine how the 2-component must be constructed by the algorithm.

(2) A unique way of establishing a 2-component still does not uniquely determine the core events, since it can happen that there is a core event in this 2-component whose basic event has trials that are also covered by other 2-components.

These two problems are the main reasons why the analysis used in the previous chapter does not directly apply here. There, the problem under consideration did not require to provide an ordering of the basic events, and a basic event corresponding to a core event cannot have trials belonging to two different 2-components. This made the proof significantly easier. In order to solve the problems above, we introduce the following structures.

The 1-Component Witness

Assume we are given a 1-component C with a set of core events $B_C = \bigcup_{i=0}^{d} E_i$, where E_0 consists of the initial core event of C and each E_i with $i \geq 1$ represents the set of core events added to C in round i of Build_1-Component. A graph $T = (B_T, E)$ is called a *1-component witness* of C if T is a directed tree with the property that

- the node set B_T of T is the set consisting of the basic events of all core events in B_C, and
- $(A, B) \in E$ for the basic events A and B if and only if
  - there is an $i \geq 0$ such that, for the core events A' and B' of A and B, $A' \in E_i$ and $B' \in E_{i+1}$, and
  - A is the event of smallest index with a core event in E_i that overlaps with B.

The 2-Component Witness

Furthermore, assume we are given a 2-component D consisting of a set of 1-components $S_D = \bigcup_{i=0}^{d} S_i$, where S_0 consists of the initial 1-component of D and each S_i with $i \geq 1$ represents the set of 1-components added to D in round i of Build_2-Component. A graph $T = (B_T \cup D_T, E_1 \cup E_2)$ is called a *2-component witness* of D if T is a tree of directed edges with the property that

- the node set B_T contains the basic events of all core events in D,
- the node set D_T of T contains the basic events of all dangerous events considered for D (see $(*)$ of Build_2-Component), which may include some of the basic events in B_T,
- $(A, B) \in E_1$ for the basic events $A, B \in B_T$ if and only if (A, B) is an edge of the 1-component witness of a 1-component in D, and
- $(A, B) \in E_2$ for the basic events $A, B \in B_T \cup D_T$ if and only if the trial sets of A and B overlap, A and B do not have core events in a single 1-component, and one of the two cases holds:
  (1) $A \in B_T$ and $B \in D_T$ and A is the event of smallest index among the basic events that had a core event in D before B was discovered to be dangerous, or
  (2) $A \in D_T$ and $B \in B_T$ and B is the event of smallest index in its 1-component, among those intersecting A, that was added to D due to A.

Edge set E_1 is called the set of *1-component edges*, and edge set E_2 is called the set of *2-component edges*.

Since according to Remark 9.5 (3) every basic event can only have one core event, 1- and 2-component witnesses are always guaranteed to be trees. Obviously, every 2-component has a unique 2-component witness. On the other hand, every 2-component witness T implies a 2-component C that is unique concerning

(1) its decomposition into 1-components,

(2) the order in which events are added to each of its 1-components in Build_1-Component,

(3) the order in which its 1-components are constructed, and

(4) the round in which each 1-component is added to C in Build_2-Component.

Item (1) follows from the fact that the 1-component witnesses in T are separated via 2-component edges. Furthermore, item (2) holds because Build_1-Component ensures that the order in which new events are added to a 1-component in one round is determined by their indices. Item (3) is true, since Step 2.1 ensures that the order in which the 1-components are constructed is determined by the indices of their initial events. Finally, item (4) holds, because Step 2.2 ensures that the initial 1-component of C is represented by the 1-component witness in T with the initial event of minimum index, and the round in which every other 1-component C' is added to C is determined by the number of 1-component witnesses that have to be passed in T from the initial 1-component witness to the 1-component witness representing C'.

However, a 2-component witness may not result in a 2-component with uniquely determined trial sets for the core events. The reason for this is that it can happen that a basic event can be a core event in one 2-component (say, C) and overlap with the trials of another 2-component (say, C'). This can happen if C' is constructed before C. Hence, in this case, a single 2-component witness is not able to uniquely specify the trials covered by the core events of a 2-component. Therefore, we also introduce 3-component witnesses.

The 3-Component Witness

Given a 2-component D, the *3-component witness* $T = (B_T \cup D_T, E_1 \cup E_2 \cup E_3)$ of D is a tree of directed edges that is iteratively constructed as follows:

Initially, T is equal to the 2-component witness of D. As long as there is some 2-component C not already in T that is in the neighborhood of an event in B_T, we add a *3-component edge* (A, B) to E_3 for an arbitrary pair of intersecting basic events A in T and B having a core event in C, and we add the 2-component witness of C to T.

Given a 3-component witness T, let V_T denote the set of all trials covered by the basic events in B_T. 3-component witnesses have the following important property.

Lemma 9.9 *Every 3-component witness uniquely determines the 2-component of its initial 2-component witness.*

Proof. As noted above, a 2-component witness already completely specifies a 2-component up to the trial sets covered by its core events. Since

- a 3-component witness provides a complete time ordering in which events are added to the 2-components it is witnessing,
- each time a core event is added to a 1-component, all trials of its basic event are taken that are not already covered by other 1-components, and
- there are no more 2-components that overlap with events in T,

a 3-component witness ensures the unique determination of the trials covered by the core events of the 2-component corresponding to its initial 2-component witness (and all other witnessed 2-components). □

This implies the following result.

Corollary 9.10 *For any 2-component C constructed by the algorithm, there is a 3-component witness that uniquely determines all core events in C.*

Corollary 9.10 implies that if there is no 3-component witness including a given set of basic events B, then B cannot form the core of a 2-component. This does significantly help to bound the size of 2-components, since instead of counting combinatorial structures based on core events, we are allowed to count combinatorial structures based on *basic* events, which is much easier. Before we do this, we introduce some notation.

For every basic event A, let $\lambda_A = |T_A|^\epsilon$ be called the *order* of A and any of its reduced events. Recall that the probability of an event A or any of its reduced events to be true is at most $p_A \le \mathrm{e}^{-\lambda_A}$ (see Theorem 9.4). The *order* of a set $\mathcal{E}$ of events is defined as $\sum_{A \in \mathcal{E}} \lambda_A$. Furthermore, for every event A in a 3-component witness T, we define the *witness order* (or *ω-order* in short) ω_A of A to be equal to λ_A if $A \in B_T$ and otherwise to some value of at least λ_A^ϵ that will be specified later. The *ω-order* ω_T of a 3-component witness T is the sum of the ω-orders of all of its events in $B_T \cup D_T$. We intend to count the expected number of 3-component witnesses T that have some given ω-order. Since $\omega_T \ge \lambda_{B_T}$, which is at least the sum of the orders of the core events of the 2-component represented by its initial 2-component witness, it holds:

Lemma 9.11 *For any 2-component C, its corresponding 3-component witness T ensures that $\omega_T \ge \lambda_{B_C}$.*

Hence, in order to bound the expected number of 2-components of order at least λ, it suffices to bound the expected number of 3-component witnesses of ω-order at least λ. This will enable us to prove Lemma 9.8.

In order to count 3-component witnesses T, we start with the basic event that represents the initial event of the initial 1-component of T. Given any set of basic events, we count the possibilities of forming 1-, 2-, and 3-component edges from the basic events to neighboring basic events. (That is, we count 3-component witnesses level-wise.) The structures we intend to use for counting all possibilities for one level of the 3-component witness are formalized in the following definition.

Definition 9.12 *Given any set of basic events $\mathcal{E}$, a set of basic events $\mathcal{F}$ is called a* witness extension *of $\mathcal{E}$ if $\mathcal{E}$ and $\mathcal{F}$ can be connected by a forest of 1-, 2-, and 3-component edges in a way that they fulfill all properties of two consecutive levels of a 3-component witness.*

Recall that the trial sets covered by the core events of a 2-component are disjoint. Therefore, the probability of a basic event to be true restricted to the trial set of its core event is independent of other events. This will be essential for estimating the expected number of witness extensions that cover a given number of nodes. To be able to apply these structures for counting 3-component witnesses, we need a technical lemma.

9.2.3 A Technical Lemma

Consider some fixed set $\mathcal{E}$ of basic events. Suppose that for every event $A \in \mathcal{E}$ we have a non-negative random variable Λ_A which we call its *contribution*. Then the *contribution* $\Lambda_{\mathcal{E}}$ of $\mathcal{E}$ is defined as the sum of the contributions of all events in $\mathcal{E}$. Let

$$S_\omega^{\mathcal{E}} = \{\text{witness extensions } \mathcal{F} \text{ of } \mathcal{E} \text{ with } \Lambda_{\mathcal{F}} = \omega\} \ .$$

The next lemma provides an estimation for $\mathrm{E}[|S_\omega^{\mathcal{E}}|]$.

Lemma 9.13 *Let γ be an arbitrary positive constant with $\gamma \leq 1/288$. Furthermore, let $\mathcal{E}$ be any set of basic events and $(\Lambda_B)_{B \in N(\mathcal{E})}$ be any sequence of contributions with the property that*

(1) the contribution of any event in $N(\mathcal{E})$ is either 0 or at least $1/\gamma$,

(2) $|\{(A, B) \in \mathcal{E} \times N(\mathcal{E}) : \ \Pr[1 \leq \Lambda_B \leq k] > 0\}| \leq \omega_{\mathcal{E}} \cdot \mathrm{e}^{\gamma \cdot k}$,

(3) for every $A \in \mathcal{E}$ and $B \in N(A)$ there are at most d different ways of linking B to A, and

(4) there is a $\phi \geq 1/6$ so that for every event $A \in N(\mathcal{E})$, $\Pr[\Lambda_A = k] \leq \mathrm{e}^{-\phi \cdot k}$ independently of other events of positive contribution.

Then

$$\mathrm{E}[|S_\omega^{\mathcal{E}}|] \leq \mathrm{e}^{-\phi\omega/2} \cdot \mathrm{e}^{d \cdot \omega_{\mathcal{E}}/48} \ .$$

Proof. Follows directly from the proof of Lemma 8.19. □

9.2.4 Counting Witness Extensions of 1- and 3-Component Edges

In this section we show how to apply Lemma 9.13 to bound the expected number of ways to select basic events of a certain order that form a witness extension of 1- and 3-component edges, called a *1/3-witness extension* in the following. For this we first need some simple claims.

Claim 9.14 *For every event A it holds*

$$\sum_{B \in N(A)} \mathrm{e}^{-\delta \cdot \lambda_B^\epsilon} \leq \lambda_A^\epsilon \ .$$

Proof. From the conditions of Theorem 9.4 and the fact that $\ln p_A^{-1} = \lambda_A$ we obtain for every event A that

$$\mathrm{e}^{-\lambda_A^\epsilon} \leq \mathrm{e}^{-\delta\lambda_A^\epsilon} \cdot \prod_{B \in N(A)} \left(1 - \mathrm{e}^{-\delta\lambda_B^\epsilon}\right) \ .$$

This implies that

$$\mathrm{e}^{-\lambda_A^\epsilon} \leq \prod_{B \in N(A)} \left(1 - \mathrm{e}^{-\delta\lambda_B^\epsilon}\right) \leq \prod_{B \in N(A)} \exp\left(-\mathrm{e}^{-\delta\lambda_B^\epsilon}\right) = \exp\left(-\sum_{B \in N(A)} \mathrm{e}^{-\delta\lambda_B^\epsilon}\right) \ ,$$

which immediately yields the claim. □

Claim 9.15 *For every event A and $k > 0$ it holds that*

$$|\{B \in N(A) : \lambda_B \leq k\}| \leq \mathrm{e}^{\delta \cdot k^\epsilon} \cdot \lambda_A^\epsilon \ .$$

Proof. Follows directly from Claim 9.14 above. □

Claim 9.16 *Every event must have an order of at least $(1/\delta)^{1/\epsilon}$.*

Proof. In Theorem 9.4 we require that $x_i \leq 1/\mathrm{e}$ for all $i \in [m]$. This implies that, for every event A_i,

$$\mathrm{e}^{-\delta \ln^\epsilon p_i^{-1}} \leq \frac{1}{\mathrm{e}} \quad \Leftrightarrow \quad \delta \cdot \lambda_{A_i}^\epsilon \geq 1 \quad \Leftrightarrow \quad \lambda_{A_i} \geq (1/\delta)^{1/\epsilon} .$$

□

Now we are ready to apply Lemma 9.13. Consider any set $\mathcal{E}$ of basic events. Let the contribution Λ_B of an event $B \in N(\mathcal{E})$ be defined either as its order λ_B if the core event corresponding to B is true, or 0 otherwise. (Recall that according to Corollary 9.10 the core event of B is uniquely determined although we only count basic events.)

Condition (1) of Lemma 9.13 follows from Claim 9.16. Our assumption that $\omega_A \geq \lambda_A^\epsilon$ for all events $A \in \mathcal{E}$ and Claim 9.15 together imply that

$$|\{(A,B) \in \mathcal{E} \times N(\mathcal{E}) : \omega_B \leq k\}| \leq \mathrm{e}^{\delta \cdot k^\epsilon} \sum_{A \in \mathcal{E}} \lambda_A^\epsilon \leq \mathrm{e}^{\delta \cdot k^\epsilon} \cdot \omega_{\mathcal{E}} .$$

Thus, condition (2) is true. For every $A \in \mathcal{E}$ and $B \in N(A)$ we have 3 choices of connecting B to A: 2 choices for a 1-component edge, and one choice for a 3-component edge. We will not count the possibilities of B being dangerous or not dangerous here, since this will be considered in the analysis for the 2-component edges. Hence, condition (3) holds with $d = 3$.

In order to fulfill condition (4), we use only a part of the probability we obtain for a core event to be true. The reason for this is that we want to reserve the other part for the 2-component edges. In particular, instead of a probability of $\mathrm{e}^{-\lambda_B}$ for the core event of B to be true, we only use a probability of $\mathrm{e}^{-\lambda_B/2} = \mathrm{e}^{-\omega_B/2}$. Thus, condition (3) follows with $\phi = 1/2$ and by the fact that the core events use disjoint trial sets.

Since all requirements of Lemma 9.13 are fulfilled, the expected number of possibilities $C_{\mathcal{E},\omega}^{(1,3)}$ of choosing events of contribution altogether ω for a 1/3-witness extension of $\mathcal{E}$ satisfies

$$C_{\mathcal{E},\omega}^{(1,3)} \leq \mathrm{e}^{-\omega/4} \cdot \mathrm{e}^{\omega_{\mathcal{E}}/16} . \tag{9.2}$$

9.2.5 Counting Witness Extensions of 2-Component Edges

In this section we show how to apply Lemma 9.13 to bound the expected number of ways to select event sets of a certain order that form a witness extension of 2-component edges, called *2-witness extension* in the following.

Let us fix some set $\mathcal{E}$ of events. We consider two different cases for counting 2-component edges of the form $(A, B) \in \mathcal{E} \times N(\mathcal{E})$ for a 3-component witness T:

(1) $B \in B_T$, or

(2) $B \in D_T \setminus B_T$.

We start with the first case. For this case, we define the contribution Λ_B of B as λ_B if the core event of B is true and 0 otherwise. Using this definition, condition (1) of Lemma 9.13 follows from Claim 9.16. Analogous to Section 9.2.4, also condition (2) is true. For every $A \in \mathcal{E}$ and $B \in N(A)$ we have 2 choices of connecting B to A: either A is dangerous, or B. Hence, condition (3) holds with $d = 2$. Furthermore, as in Section 9.2.4, condition (4) holds with $\phi = 1/2$.

Now we consider the second case. This case significantly differs from our previous considerations, since B itself does *not* provide a probability (B does not have a core event like the other events!). Therefore, we have to "borrow" probabilities from other basic events. To achieve this, we define the ω-order ω_B of the dangerous event B to be the sum of the orders of the set S of core events that were added to the 2-component due to B. From $(*)$ of Build_2-Component we know that the core events in S have to cover at least λ_B trials of B. That is, $\sum_{C\in S}\lambda_C^{1/\epsilon}\geq\lambda_B$. Since $(\sum_{C\in S}\lambda_C)^{1/\epsilon}\geq\sum_{C\in S}\lambda_C^{1/\epsilon}$, we must have $\omega_B\geq\lambda_B^{\epsilon}$. Since for each of the core events C in S we still have a probability of $\mathrm{e}^{-\lambda_C/2}$ available, we can assign a probability of $\mathrm{e}^{-\omega_B/2}\leq\mathrm{e}^{-\lambda_B^{\epsilon}/2}$ to B.

Let us define the *contribution* Λ_B of B as ω_B if the core events added to the 2-component due to B are true, and 0 otherwise. Then Claim 9.16 together with the inequality $\omega_B\geq\lambda_B^{\epsilon}$ establishes condition (1) of Lemma 9.13. For the remaining two conditions we need a simple claim.

Claim 9.17 *For every event A and every $k>0$ it holds*

$$|\{B\in N(A):\ \omega_B\leq k\}|\leq|\{B\in N(A):\lambda_B\leq k^{1/\epsilon}\}|\leq\mathrm{e}^{\delta\cdot k}\cdot\lambda_A^{\epsilon}\ .$$

Proof. The proof of the claim follows directly from Claim 9.15. □

Condition (2) follows from Claim 9.17, condition (3) follows with $d=1$, and condition (4) follows with $\phi=1/2$.

Combining the two cases (which yields $d=3$), it follows from Lemma 9.13 that the expected number of possibilities $C^{(2)}_{\mathcal{E},\omega}$ of choosing events of contribution ω for a 2-witness extension of $\mathcal{E}$ satisfies

$$C^{(2)}_{\mathcal{E},\omega}\leq\mathrm{e}^{-\omega/4}\cdot\mathrm{e}^{\omega\varepsilon/16}\ . \tag{9.3}$$

9.2.6 Counting Witness Extensions

Now we are ready to count arbitrary witness extensions. Given a set $\mathcal{E}$ of basic events, our aim is to count the expected number of witness extensions $\mathcal{F}$ of $\mathcal{E}$ of ω-order ω, denoted by $C_{\mathcal{E},\omega}$. Let $S_{\omega,\delta}$ represent the set $\{0,\lceil 1/\delta\rceil,\lceil 1/\delta\rceil+1,\ldots,w-\lceil 1/\delta\rceil,w\}$. According to the previous two sections we obtain

$$\begin{aligned}C_{\mathcal{E},\omega} &\leq \sum_{\kappa\in S_{\omega,\delta}} C^{(1,3)}_{\mathcal{E},\kappa}\cdot C^{(2)}_{\mathcal{E},\omega-\kappa}\\ &\leq \sum_{\kappa\in S_{\omega,\delta}} \mathrm{e}^{-\kappa/4}\cdot\mathrm{e}^{\omega\varepsilon/16}\cdot\mathrm{e}^{-(\omega-\kappa)/4}\cdot\mathrm{e}^{\omega\varepsilon/16}\\ &\leq \max[\omega-2/\delta+3,\ 2]\cdot\mathrm{e}^{-\omega/4}\cdot\mathrm{e}^{\omega\varepsilon/8}\leq\mathrm{e}^{-\omega/6}\cdot\mathrm{e}^{\omega\varepsilon/8}\end{aligned}$$

if $\delta>0$ is sufficiently small.

9.2.7 Proof of Lemma 9.8

Assume that we intend to bound the expected number of 2-components of order at least λ. According to Lemma 9.11, this can be achieved by bounding the expected number of 3-component witnesses of ω-order at least λ. Suppose that the 3-component witnesses have to start with some fixed initial event A_0 of ω-order ω_0. Then it holds for all $\omega\geq\omega_0$ that

$$\mathrm{E}\left[|\{\text{3-component witnesses } T \text{ with initial event } A_0 \text{ and } \omega_T=\omega\}|\right]$$

$$
\begin{aligned}
&\le \sum_{\ell=0}^{\delta\omega} \sum_{\substack{\omega_1,\ldots,\omega_\ell \in \{\lceil 1/\delta \rceil,\ldots,\omega\}: \\ \sum_{j\ge 0} i_j = \omega}} \mathrm{e}^{-\omega_0} \prod_{j=0}^{\ell-1} \mathrm{e}^{-\omega_{j+1}/6} \cdot \mathrm{e}^{\omega_j/8} \\
&\le \binom{\omega+\delta\omega}{\delta\omega} \cdot \mathrm{e}^{-\omega/6} \cdot \mathrm{e}^{\omega/8} \le \left(\frac{\mathrm{e}(1+\delta)}{\delta}\right)^{\delta\omega} \mathrm{e}^{-\omega/24} \\
&\le \mathrm{e}^{-\omega/48}
\end{aligned}
$$

if $\delta > 0$ is sufficiently small. On the other hand we know that

$$\omega_T = \sum_{A \in B_T \cup D_T} \omega_A \ge \sum_{A \in B_T} \lambda_A = \sum_{A \in B_T} |T_A|^\epsilon .$$

Since $\sum_{A\in B_T} |T_A|^\epsilon \ge (\sum_{A\in B_T} |T_A|)^\epsilon$ and $\sum_{A\in B_T} |T_A| \ge |V_T|$, it follows that $|V_T| \le \omega_T^{1/\epsilon}$.

Now we are ready to bound the number of trials covered by the 2-components after Step 2 and Step 3 of our algorithm.

Step 2:

Since there are at most m events that can be the initial event of a 3-component witness of ω-order at least ω, we obtain

$$\mathrm{E}[|\{\text{3-component witnesses } T \text{ of } \omega\text{-order at least } \omega\}|] \le m \cdot \sum_{\kappa \ge \omega} \mathrm{e}^{-\kappa/48} \le 50m \cdot \mathrm{e}^{-\omega/48} .$$

This is at most $m^{-\alpha}$ if $\omega \ge 48((\alpha+1)\ln m + 4)$. Thus, with probability at least $1 - 1/m^\alpha$ for any constant $\alpha > 0$ the number of trials covered by a 3-component witness is at most $[48((\alpha+1)\ln m + 4)]^{1/\epsilon}$, which proves part (1) of Lemma 9.8.

Step 3:

For every 2-component C covering at most $\gamma \ln^{1/\epsilon} m$ trials, its set S of core events fulfills $\sum_{A\in S} \lambda_A \le \gamma \ln^{1/\epsilon} m$. Hence, according to Claim 9.15 there can be at most $\gamma(\ln m)^{1/\epsilon} \cdot \mathrm{e}^{\delta k^\epsilon}$ events of order at most k that are participating in C. Therefore, the expected number of 3-component witnesses of order at least ω in C is at most

$$\sum_{\kappa\ge\omega} \gamma(\ln m)^{1/\epsilon} \cdot \mathrm{e}^{\delta\cdot\kappa^\epsilon} \cdot \mathrm{e}^{-\kappa/48} \le \gamma(\ln m)^{1/\epsilon} \sum_{\kappa\ge\omega} \mathrm{e}^{-\kappa/96} \le 100\gamma(\ln m)^{1/\epsilon} \cdot \mathrm{e}^{-\omega/96}$$

if ϵ and δ are sufficiently small. This is at most $(\ln m)^{-\alpha}$ if $\omega \ge 96((\alpha+1/\epsilon)\ln\ln m + \ln(100\gamma))$. Thus with probability at least $1 - 1/(\ln m)^\alpha$ for any constant $\alpha > 0$ the number of trials covered by a 3-component witness is at most $[96((\alpha+1/\epsilon)\ln\ln m + \ln(100\gamma))]^{1/\epsilon}$, which proves part (2) of Lemma 9.8.

9.3 The MIP Algorithm

In this section we present an algorithm that fulfills Theorem 9.3. First we give a high-level description of the algorithm.

Given an arbitrary MIP, we start with finding an optimal solution $\{x^*_{i,j} : \ i \in [n],\ j \in [\ell_i]\}$ to the LP relaxation of the MIP (i.e., the integrality constraints in Definition 9.1 (3) are removed).

The resulting LP optimum $\boldsymbol{y}^*$ is clearly a lower bound for $\boldsymbol{y}$, the optimum of the (integral form of the) MIP.

Afterwards, our algorithm works in 3 steps. Its structure is similar to the algorithm in [LRS99], but the techniques used here are completely different. Step 1 rounds the $x^*_{i,j}$'s to multiples of some number in $\Omega(1/\ln m)$. Step 2, which contains the main novel contribution in this chapter, rounds the $x^*_{i,j}$'s until all of them are above some constant $\gamma < 1$ or 0. Step 3 finally rounds the $x^*_{i,j}$'s to values in $\{0, 1\}$.

Step 1: Initial Rounding

For every $r \in \{1, \ldots, m\}$, let $w_r = \max_c a_{r,c}$. First, we scale $\boldsymbol{A}$ to a matrix $\boldsymbol{A}^{(1)} = (a^{(1)}_{r,c})$ by multiplying row r of $\boldsymbol{A}$ with w_r^{-1} for all r. This ensures that $\max_c a^{(1)}_{r,c} = 1$ for all r. Let the vector $\boldsymbol{y}^{(1)}$ be equal to $\boldsymbol{A}^{(1)}\boldsymbol{x}^*$. For the remaining steps we will assume that the approximation vector $\boldsymbol{\alpha}$ fulfills the following property for some $0 < \epsilon < 1$ (see Theorem 9.3):

$$\alpha_r \geq \sigma_r \quad \text{with} \quad \sigma_r = \max\left[(y_r^{(1)})^{-1},\ (y_r^{(1)})^{-(1-\epsilon)/2}\right] .$$

Let $\mu = \max_r \lceil (6 \ln m)/(\min[\sigma_r, \sigma_r^2] y_r^{(1)}) \rceil$. Since $\min[\sigma_r, \sigma_r^2] \cdot y_r^{(1)} \geq 1$ for every r, $\mu \leq \lceil 6 \ln m \rceil$. If $\mu = 1$, then we simply use the randomized rounding strategy by Raghavan and Thompson [RT87] to transform $\boldsymbol{x}^*$ into an integral vector $\boldsymbol{x}^{(1)}$. That is, for every $i \in [n]$ we set the variable $x^{(1)}_{i,j}$ to 1 and the other $x^{(1)}_{i,j'}$'s to 0 with probability $x^*_{i,j}$. In this case,

$$\begin{aligned}\Pr[(\boldsymbol{A}^{(1)}\boldsymbol{x}^{(1)})_r \geq (1+\alpha_r)y_r^{(1)}] &\leq \mathrm{e}^{-\min[\alpha_r,\alpha_r^2]y_r^{(1)}/3} \\ &\leq \mathrm{e}^{-\min[\sigma_r,\sigma_r^2]y_r^{(1)}/3} \leq \mathrm{e}^{-2\ln m} = \frac{1}{m^2}\end{aligned}$$

for every row r. Hence, with high probability we obtain an integral solution $\boldsymbol{x}^{(1)}$ with $(\boldsymbol{A}^{(1)}\boldsymbol{x}^{(1)})_r < (1+\alpha_r)y_r^{(1)}$ for all r as proposed in Theorem 9.3.

For the case that $\mu > 1$, we select a matrix $\boldsymbol{R} = (r_{i,k})_{i \in [n], k \in [\mu]}$ uniformly at random out of $[0,1)^{n \times \mu}$. For every i, j, we set $x^{(1)}_{i,j} = |\{k :\ r_{i,k} \in [\sum_{\ell=1}^{j-1} x^*_{i,\ell}, \sum_{\ell=1}^{j} x^*_{i,\ell})\}|/\mu$. Consider a system of bad events $E_1, \ldots, E_m$ with E_r being true if and only if $(\boldsymbol{A}^{(1)}\boldsymbol{x}^{(1)})_r \geq (1+\alpha_r)y_r^{(1)}$. We show now how to bound the probability of E_r being true for any r. For any $i \in [n]$ and $k \in [\mu]$, let the random variable $X_{i,k}$ be equal to $a^{(1)}_{r,(i,j)}/\mu$ if and only if $r_{i,k}$ is selected so that it contributes $1/\mu$ to $x^{(1)}_{i,j}$. The definition ensures that $\sum_{i,k} X_{i,k} = (\boldsymbol{A}^{(1)}\boldsymbol{x}^{(1)})_r$. Obviously,

$$\mathrm{E}[\sum_{i,k} X_{i,k}] = \mathrm{E}[(\boldsymbol{A}^{(1)}\boldsymbol{x}^{(1)})_r] = y_r^{(1)} .$$

Since $X_{i,k} \leq 1/\mu$ for all i, k and the random variables are independent, we can apply the Hoeffding bounds (after a scaling) to obtain

$$\Pr[E_r] \ \leq\ \mathrm{e}^{-\min[\alpha_r,\alpha_r^2]\cdot\mu\cdot y_r^{(1)}/3} \leq \mathrm{e}^{-2\ln m} = \frac{1}{m^2} .$$

for all r. Hence, with high probability none of the bad events occurs. Assume that the random experiment has been successful in avoiding all bad events (otherwise we repeat it). Then we replace $\boldsymbol{x}^*$ by the N-vector $\boldsymbol{x}^{(1)}$. In order to simplify the following calculations, we assume in the following as a worst case that, after Step 1, $(\boldsymbol{A}^{(1)}\boldsymbol{x}^{(1)})_r = (1+\alpha_i)y_r^{(1)}$ for all r.

Step 2: Intermediate Rounding

This is the first step that requires the LLL (and the consideration of a dependency graph that may have already been significantly reduced by the previous step). The aim of this step is to round the vector $\boldsymbol{x}^{(1)}$ with entries in $\{k/\mu :\ k \in \{0,\dots,\mu\}\}$ to a vector $\boldsymbol{x}^{(2)}$ with entries in $\{k/\gamma :\ k \in \{0,\dots,\gamma\}\}$ for some fixed integer $\gamma > 1$. This will be done in several phases, starting with phase 1. Initially, we set $\boldsymbol{z}^{(0)}$ equal to $\boldsymbol{x}^{(1)}$ and μ_0 equal to μ. The outcome vector of phase $\varphi \geq 1$ is defined by $\boldsymbol{z}^{(\varphi)}$. The final vector $\boldsymbol{z}^{(\varphi)}$ of Step 2 is set to $\boldsymbol{x}^{(2)}$.

The task of phase φ is to round a vector $\boldsymbol{z}^{(\varphi-1)}$ of values that are multiples of $1/\mu_{\varphi-1}$ to a vector $\boldsymbol{z}^{(\varphi)}$ of values that are multiples of $1/\mu_\varphi$ with $\mu_\varphi = \lceil \mu_{\varphi-1}^{\epsilon} \ln \mu_{\varphi-1} \rceil$. This is done as follows.

For simplicity, set $\boldsymbol{x} = \boldsymbol{z}^{(\varphi-1)}$, $\boldsymbol{x}' = \boldsymbol{z}^{(\varphi)}$, $\mu = \mu_{\varphi-1}$, $\mu' = \mu_\varphi$, and $\boldsymbol{A} = \boldsymbol{A}^{(1)}$. The random experiment for applying the LLL is to select a matrix $\boldsymbol{R} = (r_{i,k})_{i\in[n],k\in[\mu']}$ uniformly at random out of $[0,1)^{n\times\mu'}$. For every i,j, we set $x'_{i,j} = |\{k :\ r_{i,k} \in [\sum_{\ell=1}^{j-1} x_{i,\ell}, \sum_{\ell=1}^{j} x_{i,\ell})\}|/\mu'$. Consider a system of bad events $E_1,\dots,E_m$ with E_r being true if and only if

$$(\boldsymbol{A}\boldsymbol{x}')_r > \left(1 + \frac{\min[\alpha_r,\ 1]}{\sqrt{\ln \mu}}\right) \cdot (\boldsymbol{A}\boldsymbol{x})_r\ .$$

Let the dependency graph G of the E_r's be defined as in Definition 9.2. Clearly, every event E_r is mutually independent of all events it is not connected with in G. Assume that the approximation vector $\boldsymbol{\alpha}$ is chosen such that inequality (9.1) in Theorem 9.3 holds. As will be shown in the next subsection, in this case Theorem 9.4 predicts that a vector $\boldsymbol{x}'$ can be provided in polynomial time so that for all r,

$$(\boldsymbol{A}\boldsymbol{x}')_r \leq \left(1 + \frac{c\cdot\min[\alpha_r,\ 1]}{\sqrt{\ln \mu}}\right) \cdot (\boldsymbol{A}\boldsymbol{x})_r$$

for some constant $c > 0$.

Step 2 ends with the first μ_φ for which $\mu_{\varphi+1} > \mu_\varphi^{\epsilon+(1-\epsilon)/2}$ (i.e., phase φ is the last phase). In this case, $\mu_\varphi^{(1-\epsilon)/2} < \ln \mu_\varphi$. In general, for all $0 < \delta < 1$ it holds that if $x^\delta < \ln x$ then $x < \mathrm{e}^{1/(1-\delta)}$. Thus, in our case, $\mu_\varphi < \mathrm{e}^{2/(1+\epsilon)}$, which is a constant. For all phases φ of Step 2 it therefore holds that $\mu_{\varphi-1} \geq \mu_\varphi^{2/(1+\epsilon)}$. This enables us to bound the approximation factor obtained in Step 2. We distinguish between two cases. If $\alpha_r < 1$ for some row r then

$$\begin{aligned}
\prod_{j=0}^{\infty}\left(1 + \frac{c\cdot\alpha_r}{\sqrt{\ln \mu_\varphi^{(2/(1+\epsilon))^j}}}\right) &\leq \exp\left(c\cdot\alpha_r \sum_{j=0}^{\infty} \frac{1}{(2/(1+\epsilon))^{j/2}\sqrt{\ln \mu_\varphi}}\right)\\
&\leq \exp\left(O(\alpha_r)\right) = (1 + O(\alpha_r))\ .
\end{aligned}$$

Thus, $(\boldsymbol{A}\boldsymbol{z}^{(\varphi)})_r \leq (1 + O(\alpha_r)) y_r^{(1)}$ for all r. If $\alpha_r \geq 1$, then we also obtain an approximation ratio of $(1 + O(\alpha_r))$, since $\prod_j (1 + c/\sqrt{\ln \mu_j})$ is a constant.

Applying Theorem 9.4 to Step 2

Our aim is to show that every phase of Step 2 can be made to match the requirements of Theorem 9.4. Let $\boldsymbol{x}$, $\boldsymbol{x}'$, μ, and μ' be chosen as above, and let $\boldsymbol{A}$ represent the matrix $\boldsymbol{A}^{(1)}$. In order to avoid dealing with events of non-uniform weights, we use the following strategy. Let ν be the minimum value of an $a_{i,j}$ in $\boldsymbol{A}$. For every $k \in \{0,\dots,\lfloor \log \nu \rfloor\}$ and every row r in $\boldsymbol{A}$, let the set $M_{r,k}$ contain all (i,j) with $a_{r,(i,j)} \in (2^{-(k+1)}, 2^{-k}]$ and $x_{i,j} > 0$. Set $m_{r,k} = 2^{k+1}\mu \cdot (\boldsymbol{A}\boldsymbol{x})_r$.

Obviously, $|M_{r,k}| \le m_{r,k}$. Let the matrix $\boldsymbol{B}^{(k)} = (b^{(k)}_{r,c})$ represent $\boldsymbol{A}$ with zeroes at all entries $a_{r,c}$ with $c \notin M_{r,k}$. Furthermore, let $y_{r,k}$ be chosen so that $(\boldsymbol{B}^{(k)}\boldsymbol{x})_r = y_{r,k}$. Set $y_r = \sum_{k\ge 0} y_{r,k}$. We define the event $E_{r,k}$ to be true if and only if

$$(\boldsymbol{B}^{(k)}\boldsymbol{x}')_r > y_{r,k} + \frac{6\min[\alpha_r,\ 1]}{\sqrt{2^{(1-\epsilon)k}\ln\mu}} \cdot y_r\ .$$

Assume that all events $E_{r,k}$ were false. Then it follows that

$$(\boldsymbol{A}\boldsymbol{x}')_r \le \sum_{k\ge 0}\left(y_{r,k} + \frac{6\min[\alpha_r,\ 1]}{\sqrt{2^{(1-\epsilon)k}\ln\mu}} \cdot y_r\right) = \left(1 + O\left(\frac{\min[\alpha_r,\ 1]}{\sqrt{\ln\mu}}\right)\right) y_r\ ,$$

as desired.

We show now how to bound the probability of $E_{r,k}$ being true. Recall the random experiment above of selecting the $\boldsymbol{x}'_{i,j}$'s. For any $i \in [n]$ and $\ell \in [\mu']$, let the random variable $X_{i,\ell}$ be equal to $b^{(k)}_{r,(i,j)}/\mu'$ if and only if $r_{i,\ell}$ is selected so that it contributes $1/\mu'$ to $x'_{i,j}$. The definition ensures that $\sum_{i,\ell} X_{i,\ell} = (\boldsymbol{B}^{(k)}\boldsymbol{x}')_r$. Obviously,

$$\mathrm{E}[\sum_{i,\ell} X_{i,\ell}] = \mathrm{E}[(\boldsymbol{B}^{(k)}\boldsymbol{x}')_r] = y_{r,k}\ .$$

Since $X_{i,\ell} \le 1/(\mu' 2^k)$ for all i, k and the random variables are independent, and since $y_{r,k} \le y_r$, the Hoeffding bounds give (after a scaling) that $\Pr[E_{r,k}]$ is at most

$$\exp\left(-6\left(\frac{\min[\alpha_r,\ 1]}{\sqrt{2^{(1-\epsilon)k}\ln\mu}}\right)^2 \cdot (\mu' 2^k)\cdot y_r/3\right) \le \exp\left(-2\cdot 2^{\epsilon k}\mu^{\epsilon}\min[\alpha_r^2,\ 1]y_r\right)\ .$$

Recall that $m_{r,k} = 2^{k+1}\mu\cdot y_r$ and, by our assumption, $y_r \ge (1+\alpha_r)y_r^{(1)} \ge 1$. As $\alpha_r < 1$ is only allowed for $y_r^{(1)} \ge 1$, we obtain for $\alpha_r < 1$ that $\alpha_r^2 y_r \ge y_r^{\epsilon}$ and therefore

$$2^{\epsilon k}\mu^{\epsilon}\min[\alpha_r^2,\ 1]y_r \ge 2^{\epsilon k}\mu^{\epsilon}y_r^{\epsilon} \ge m_{r,k}^{\epsilon}\cdot 2^{-\epsilon}\ .$$

Clearly, the same inequality holds for $\alpha_r \ge 1$. Thus, we have $\Pr[E_{r,k}] \le \mathrm{e}^{-m_{r,k}^{\epsilon}}$. On the other hand, the minimum number of $X_{i,\ell}$'s and therefore of $r_{i,\ell}$'s necessary to obtain a value of $6\min[\alpha_r,\ 1]/(\sqrt{2^{(1-\epsilon)k}\ln\mu})\cdot y_r$ is at least $6\cdot 2^{\epsilon k}\mu^{\epsilon}\min[\alpha_r,\ 1]y_r$, which is more than $m_{r,k}^{\epsilon}$.

Next we show how to transform inequality (9.1) in Theorem 9.3 into a condition that matches the requirements of Theorem 9.4. First, we define the dependency graph. Suppose that we are dealing with a MIP instance I. Then we define the dependency graph G'_I to contain the edge $((r,k),(s,\ell))$ if and only if $(r,s) \in G_I$ and either $r \ne s$ or $k \ne \ell$. (Here we use the fact that G_I may have self-loops.) It is easy to check that G'_I covers all dependencies among the $E_{r,k}$ events.

Let p_r, q_r, and z_r be defined as in Theorem 9.3. That is,

$$p_r = \mathrm{e}^{-\min[\alpha_r,\alpha_r^2]y_r^{(1)}/3}\ ,\quad q_r = p_r^{-\ln^{\epsilon} p_r^{-1}}\ ,\ \text{and}\ z_r = q_r^{\delta}\ .$$

Set $p_{r,k} = \exp(-2^{\epsilon k+1}\min[\alpha_r^2, 1]y_r)$. Since we assumed at the end of Step 1 that $y_r \ge (1+\alpha_r)y_r^{(1)}$, it holds that $p_{r,k} \le p_r^{2^{\epsilon k}}$. Let $q_{r,k} = \mathrm{e}^{-\ln^{\epsilon} p_{r,k}^{-1}}$ and $z_{r,k} = q_{r,k}^{\delta}$. Then we have

$$q_{r,k} = \mathrm{e}^{-\ln^{\epsilon} p_{r,k}^{-1}} \le \mathrm{e}^{-\delta 2^{\epsilon^2 k}\ln^{\epsilon} p_r^{-1}} = q_r^{2^{\epsilon^2 k}}\ . \tag{9.4}$$

Since $z_r \leq 1/3$ by Theorem 9.3, $\sum_{k\geq 0} z_r^{2^{\epsilon^2 k}} \leq \epsilon^{-2} z_r/(1-z_r) \leq 3/(2\epsilon^2)\cdot z_r$ for all r. Thus,

$$\prod_{k\geq 0} \mathrm{e}^{-z_{r,k}} \geq \mathrm{e}^{-3/(2\epsilon^2)\cdot z_r} \geq (1-z_r)^{3/(2\epsilon^2)} .$$

On the other hand, $(1-z_{r,k}) \geq \mathrm{e}^{-(4/3)z_{r,k}}$ for all $z_{r,k}$ since $\mathrm{e}^{-(4/3)x} \leq 1-4x/3+8x^2/9 \leq 1-x$ for all $x \leq 1/3$. Hence, $\prod_{k\geq 0}(1-z_{r,k}) \geq (1-z_r)^{2/\epsilon^2}$. Furthermore, due to (9.4) it holds for all (i,k) that

$$\frac{q_{r,k}}{z_{r,k}} = q_{r,k}^{1-\delta} \leq q_r^{1-\delta} = \frac{q_r}{z_r} .$$

Thus, we obtain from inequality (9.1) in Theorem 9.3 that

$$q_{r,k} \leq z_{r,k} \prod_{((r,k),(s,\ell))\in G_I'} (1-z_{s,\ell}) .$$

Next we define the trials and events used in Theorem 9.4. The $r_{i,\ell}$'s represent the trials $t_1,\ldots,t_n$, and the $E_{r,k}$'s represent the events $A_1,\ldots,A_m$. Our random experiment ensures that the trials are mutually independent. Let $T_{r,k}$ represent the set of all $r_{i,\ell}$'s for which there is an $x'_{i,j}$ counting for $E_{r,k}$ (that is, $x_{i,j} > 0$). Obviously, every event $E_{r,k}$ only depends on the outcome of the trials represented by $T_{r,k}$. If a restricted set S of $r_{i,\ell}$'s is applied to an event $E_{r,k}$, then we define $E_{r,k}|_S$ to be true if and only if

$$(\boldsymbol{B}^{(k)}\cdot\boldsymbol{x}')_r|_S > \mathrm{E}[(\boldsymbol{B}^{(k)}\cdot\boldsymbol{x}')_r|_S] + \frac{6\min[\alpha_r,\ 1]}{\sqrt{2^{(1-\epsilon)k}\ln\mu}}\cdot y_r .$$

($(\boldsymbol{B}^{(k)}\cdot\boldsymbol{x}')_r|_S$ counts only those entries that can be greater than 0 due to some $r_{i,\ell}\in S$.) This ensures that, under the assumption that an event $E_{r,k}$ can be decomposed into at most some constant number of events $E_{r,k}|_S$ that are false (as claimed by Theorem 9.4), we obtain the desired asymptotic approximation ratio. As shown above, $p_{r,k} \leq \mathrm{e}^{-m_{r,k}^{\epsilon}}$. Furthermore, we already showed that if $|S| > m_{r,k}^{\epsilon}$ then the above definition yields $\Pr[E_{r,k}|_S] \leq p_{r,k}$. Otherwise, we can set $\Pr[E_{r,k}|_S]=0$, since more than $m_{r,k}^{\epsilon}$ trials are needed to violate our deviation bound.

There is one aspect of our application that does not directly match the requirements of Theorem 9.4: $|T_{r,k}|$, the number of trials counting for $E_{r,k}$, cannot be upper bounded by $m_{r,k}$, but only by $\mu\cdot m_{r,k}$. This changes the number of trials covered by a 2-component C from at most $\sum_{E_{r,k}\in B_C} m_{r,k}$ to at most $\sum_{E_{r,k}\in B_C}|T_{r,k}| \leq \mu\sum_{E_{r,k}\in B_C} m_{r,k}$. Thus, the size of the final 2-components changes to $O(\mu(\frac{1}{\epsilon}\ln\ln m)^{1/\epsilon})$. If $\mu \geq \sqrt{\log m}$, it may not be possible any more to find a final solution via exhaustive search in polynomial time. However, every event participating in this 2-component has a probability of at most $p_{r,k} \leq \mathrm{e}^{-\mu^{\epsilon}}$ to be true. Furthermore, it follows from Claim 9.15 that a 2-component of q nodes can have at most $q\cdot\mathrm{e}^{\delta\cdot k^{\epsilon}}$ events of order at most k, and each of these events has a probability of at most e^{-k} to be true. Hence, an expected constant number of random experiments suffice for each 2-component to obtain a final solution. Thus, Step 2 of the MIP algorithm can be done in polynomial time.

Step 3: Final Rounding

Recall that $\boldsymbol{x}^{(2)}$ is the vector resulting from $\boldsymbol{x}^{(1)}$ in Step 2. Let $\boldsymbol{B}^{(k)}$, $y_{r,k}$, and y_r be defined as in the previous step. Step 3 works like a phase in Step 2 with the difference that now the random experiment simply consists of rounding $\boldsymbol{x}^{(2)}$ to an integral vector $\boldsymbol{x}$ in a way that for every i,

$x_{i,j}$ is set to 1 and the other $x_{i,j'}$'s to 0 with probability $x_{i,j}^{(2)}$. Assuming that $y_r \geq (1+\alpha_r)y_r^{(1)}$ for all r it holds

$$\begin{aligned}\Pr\left[(\boldsymbol{B}^{(k)}\boldsymbol{x})_r \geq y_{r,k} + \frac{c \cdot \min[\alpha_r, 1]}{\sqrt{2^{(1-\epsilon)k}\ln \mu_\varphi}} \cdot y_r\right] &\leq \exp\left(-c\left(\frac{\min[\alpha_r, 1]}{\sqrt{2^{(1-\epsilon)k}\ln \mu_\varphi}}\right)^2 \cdot 2^k \cdot y_r/3\right) \\ &\leq \exp\left(-2 \cdot 2^{\epsilon k}\mu_\varphi^\epsilon \min[\alpha_r^2, 1]y_r\right)\end{aligned}$$

if $c \geq 6\mu_\varphi^\epsilon \ln \mu_\varphi$, which is a constant. Hence, as in Step 2, we can apply Theorem 9.4 to obtain altogether an approximation ratio of $1 + O(\alpha_r)$ for every row r. This concludes the proof of Theorem 9.3.

9.4 Applications

In this section we present two applications of our method: job shop scheduling and MAX SAT.

9.4.1 Job Shop Scheduling

In acyclic job shop scheduling problems there are n jobs and m machines. Each job is composed of a sequence of operations to be performed on different machines. A *legal* schedule is one in which within each job, operations are carried out in order, and each machine performs at most one operation in any unit of time. If D denotes the length of the longest job and C denotes the number of time units requested by all jobs on the most loaded machine, then clearly $lb = \max[C, D]$ is a lower bound on the length of the shortest legal schedule. Let P denote the longest operation a job has on a machine.

In Chapter 7 several non-constructive results are shown about the makespan of job shop schedules. Our techniques allow us to find in polynomial time schedules with a makespan that is close to the predicted makespan (instead of 1, we obtain a $1+\epsilon$ in the exponent of some terms). In particular, we will show that the following result holds.

Theorem 9.18 *In polynomial time, acyclic job shop schedules can be computed with the following upper bounds on the makespan for any constant $\epsilon > 0$:*

(1) For any acyclic JSS problem,

$$L = O\left((C + D(\log\log P)^{1+\epsilon})^{\frac{\log P}{\log(\min[C/D + (\log\log P)^{1+\epsilon}, P])}}\right),$$

implying that $L = O(lb \log lb (\log\log lb)^{1+\epsilon})$. Observe that if $C \geq D \cdot P^\delta$ for some constant $\delta > 0$, then $L = \Theta(C)$.

(2) If operation lengths depend only on the machine on which the operation is performed, then $L = O(C + D(\log\log P)^{1+\epsilon})$.

(3) With preemption, $L = O(C + D(\log\log P)^{1+\epsilon})$.

In order to prove Theorem 9.18, we have to constructivize the existence proof given for an intermediate schedule in Section 7.2.4. Assume that we are given a fixed constant $0 < \epsilon < 1$. Let $d = \log^{1/\epsilon} D$. To avoid dealing with floors and ceilings we assume in the following that all fractions and logarithms result in integer values. It is easy to check that this does not affect our asymptotic bounds. As a further simplification, we assume w.l.o.g. that $D \leq P^3$ and $C \leq P^3 \cdot d$. If not, this can be achieved in the following way:

We use a sequence of refinements, starting with an initial (illegal) schedule S_0 in which every job is executed at every time step until it is completed. Hence, the length L_0 of S_0 is at most D. First, we expand S_0 to a schedule of length at most $2D$ in which every operation is contained completely in some time interval I of length P, starting and ending at an integer multiple of P. In the following, we assume all delays and splits to be integral multiples of P. (This will simplify partitioning a schedule into subschedules.) The symmetric LLL predicts that suitable delays can be chosen for the jobs out of a range of $[C/d]$ so that, for $\ln C \geq P$, the congestion within any time interval of size $\ln^3 C$ is at most $(1 + O(1/\sqrt{\ln C}))d\ln^3 C$. We split such a schedule into subschedules of length $L_1 = \ln^4 C$ and continue to refine each of these subschedules independently. The LLL predicts that for each of these subschedules there are delays for the jobs in the range $[L_1^{3/4}]$ so that, for $\ln L_1 \geq P$, the congestion within any time interval of size $\ln^3 L_1$ is at most $(1 + O(1/\sqrt{\ln L_1}))d\ln^3 L_1$. Again, we split such a schedule into subschedules of length $L_2 = \ln^4 L_1$. The refinements can be continued until we reach schedules of length P^3 in which the congestion is at most $O(P^3 \cdot d)$. (To achieve a congestion of at most $P^3 \cdot d$, another application of the symmetric LLL partitions these subschedules into subschedules of length P^3 with congestion at most $P^3 \cdot d$.) In order to find these subschedules in polynomial time, the algorithm by Leighton, Maggs, and Richa [LMR99] can be used. Each of these subschedules will then be refined independently by the methods presented below.

For every $i \in \{1, \ldots, \log_d D\}$, we define the set T_i as $\{D/d^i, \ldots, D/d^{i-1}-1\}$. An operation of a job is called a T_i*-operation* if its length is in T_i. For any time interval I and schedule S, let $S|_I$ denote S restricted to I (that is, $S|_I$ only contains operations that start in I). Fix some machine M and schedule S. Then the T_i*-congestion* $C_i^S(M)$ at M is defined as the congestion at M caused by T_i-operations in S. Similarly, the T_i*-contention* $c_i^S(M)$ at M is defined as the contention at M caused by T_i-operations in S. Furthermore, given any fixed time step t, $c_i^S(M,t)$ denotes the contention at M caused by T_i-operations in step t of schedule S.

Our strategy will be to refine some schedule S to some schedule S'. If we are already at the rth refinement step, $r \geq 1$, this is done by cutting S into consecutive time frames of length $D/d^{\max[r-2,0]}$. Each operation is associated with the frame in which it starts. Within each frame, each job is given a suitable delay in the range $[D/d^{r-1} - D/d^r]$. Thus, a job starting at time t in some frame F that chooses a delay of δ now starts at time $t+\delta$ in F. Let S_F be the schedule for F before inserting the delays, and let S'_F be the schedule for F afterwards. Our goal is to find delays for the jobs so that the congestion and/or the contention within certain time intervals of S'_F is sufficiently balanced for every group of T_i-operations. In order to specify the congestion and contention bounds we want to achieve, we need some notation.

For any $j \geq 0$, let $\mathcal{I}_j$ denote the set of all possible time intervals of length D/d^j in the time range of S'_F, and let $\ell_i = D/d^{i-1}$ denote an upper bound on the largest possible length of a T_i-operation. Given a fixed machine M, let $\hat{C}_i(M) = \max_{I \in \mathcal{I}_{r-1}} C_i^{S_F|_I}(M)$. Furthermore, given a fixed machine M and fixed time interval $I \in \mathcal{I}_r$, let $C'_i(M,I) = C_i^{S'_F|_I}(M)$ and $\hat{C}'_i(M) = \max_{I \in \mathcal{I}_r} C'_i(M,I)$. W.l.o.g. we may assume that $\hat{C}_i(M)$ is at most $\ell_i \cdot d^k$ for all M, where $k = 1+1/\epsilon$. Otherwise, we separate the operations into different sets of congestion in $[\ell_i \cdot d^k/2, \ell_i \cdot d^k]$. Since all T_i-operations have a length of less than ℓ_i, this is always possible.

Since we assume that $D \leq P^3$, Theorem 9.18 follows from the following lemma (which replaces Claims 7.11 and 7.13).

Lemma 9.19 *For any constant* $0 < \epsilon < 1$*, a schedule* S'_F *can be found in polynomial time so that the following conditions hold for* S'_F *for every time frame* F.

For every $i \geq r+2$, every machine M and time interval $I \in \mathcal{I}_r$:

(1) If $\hat{C}_i(M) = \ell_i \cdot d^k$ for some $2 \leq k \leq 1+1/\epsilon$, then $\hat{C}'_i(M,I) = (1+O(\sqrt{d^{-(k-2)}}))\hat{C}_i(M)/(d-1)$.

(2) If $\hat{C}_i(M) < \ell_i \cdot d^2$, then $\hat{C}_i(M,I) = O(\ell_i \cdot d)$.

Furthermore, for every machine M and time step t in S'_F:

(3) $c_{r+1}^{S'_F}(M,t) = O(C/D + (\log\log D)^{1/\epsilon})$, and

(4) $c_r^{S'_F}(M,t) = O(c_r^{S_F})$.

Proof. For every machine M and time interval $I \in \mathcal{I}_r$, we define the event $A_{M,i,I}$ to be true if and only if

$$\hat{C}'_i(M,I) \geq \begin{cases} (1+b_1\sqrt{d^{-(k-2)}})\hat{C}_i(M)/(d-1) & : \quad k \geq 2 \\ b_1 \cdot \ell_i \cdot d & : \quad k < 2 \end{cases}$$

for some constant $b_1 > 0$ to be fixed later. Moreover, for every machine M and time step t in S'_F, we define the event $B_{M,i,t}$ to be true if and only if

$$c_i^{S'_F}(M,t) \geq \begin{cases} b_2(C/D + (\log\log D)^{1/\epsilon}) & : \quad i = r+1 \\ b_2 \cdot c_r^{S_F} & : \quad i = r \end{cases}$$

for some constant $b_2 > 0$ to be fixed later.

First, we prove probability bounds for the events and bound the dependencies among them. Afterwards, we show that the events fulfill all requirements of Theorem 9.3.

Bounding the probabilities

We start with the congestion events. First, we bound the probability that item (1) is not fulfilled. Since the jobs choose their delays uniformly at random out of the range $[D(d-1)/d^r]$, the expected value of $\hat{C}'_i(M,I)$ is at most

$$\frac{D}{d^r} \cdot \frac{\hat{C}_i(M)}{D(d-1)/d^r} = \frac{\hat{C}_i(M)}{d-1} .$$

As the operations considered in $\hat{C}_i(M)$ have a length of less than ℓ_i and the jobs choose their delays independently at random, the Hoeffding bounds imply that, for any fixed machine M and time interval $I \in \mathcal{I}_r$,

$$\begin{aligned} \Pr[\hat{C}'_i(M,I) \geq (1+b_1\sqrt{d^{-(k-2)}})\hat{C}_i(M)/(d-1)] &\leq \mathrm{e}^{-b_1 \cdot d^{-(k-2)} d^{k-1}/3} \\ &= \mathrm{e}^{-b_1 \cdot d/3} \end{aligned}$$

for any constant $b_1 \geq 1$. For item (2), observe that the probability that (2) is not fulfilled can also be made to be at most $\mathrm{e}^{-b_1 \cdot d/3}$ for any constant $b_1 \geq 1$.

Next we consider the contention events. We start with item (3). Consider any fixed machine M and any fixed time step t in S'_F. According to Section 7.2.4, if the congestion events hold for every refinement step, then the expected value of $c_{r+1}^{S'_F}$ is at most $\gamma \cdot C/D$ for some constant

$\gamma \geq 1$ (recall that we assume $C \geq D$). Since the jobs choose their delays independently at random, we obtain

$$\Pr[c_{r+1}^{S'_F}(M,t) \geq b_2(C/D + (\log\log D)^{1/\epsilon})] \quad \leq \quad \mathrm{e}^{-(b_2/\gamma - 1)(C/D + (\log\log D)^{1/\epsilon})/3}$$

for any constant $b_2 \geq 2\gamma$.

For item (4), note that the expected value for $c_r^{S'_F}$ is at most $\gamma'(C/D + (\log\log D)^{1/\epsilon})$ if item (3) is always fulfilled. Thus,

$$\Pr[c_r^{S'_F}(M,t) \geq b_3(C/D + (\log\log D)^{1/\epsilon})] \quad \leq \quad \mathrm{e}^{-(b_3/\gamma' - 1)(C/D + (\log\log D)^{1/\epsilon})/3}$$

for any constant $b_3 \geq 2\gamma'$. Hence, we can achieve the same probability as in case (3).

Bounding the dependencies

Whether or not a bad event $A_{M,i,I}$ happens only depends on the delays assigned to the jobs. Hence, two bad events $A_{M,i,I}$ and $A_{M',i',I'}$ are independent unless some job has operations that are considered both for $A_{M,i,I}$ and for $A_{M',i',I'}$. According to our assumptions, at most $d^{2+1/\epsilon}$ jobs are considered for any $A_{M,i,I}$. Each of these jobs can have operations on at most D other machines, and there are at most $2D$ time intervals in $I \in \mathcal{I}_r$ to be considered per machine. Hence, any event $A_{M,i,I}$ depends on at most $d^{2+1/\epsilon} \cdot 2D^2$ other events $A_{M',i',I'}$.

Next we consider the contention events. Two bad events $B_{M,i,t}$ and $B_{M',i',t'}$, $i, i' \in \{r, r+1\}$, are independent unless some job has operations that count both for $B_{M,i,t}$ and for $B_{M',i',t'}$. To keep the number of dependent events as low as possible, let us restrict the number of time steps to be considered in S'_F to those that are integer multiples of D/d^{r+1}. Suppose that we know that, for some machine M and $i \in \{r, r+1\}$, the maximum contention at any of these time steps is c. Then the maximum contention for M and i over all time steps of S'_F can be at most $2c$, since any operation considered for $B_{M,i,t}$ has a length of at least D/d^{r+1}. Since the length of F is $D/d^{\max[r-2,0]}$ and the delay range is less than D/d^{r-1}, at most $2d^3$ time steps have to be considered for each M and i. According to our assumption that $C \leq D \cdot d$, the contention in S_F is at most $O(d)$ (assuming that all events have been true in previous refinement steps). Thus, at most $O(d) \cdot (D/d^{r-1})/(D/d^{r+1}) = O(d^3)$ operations are considered for each $B_{M,i,t}$. Since a job can have at most $(D/d^{\max[r-2,0]})/(D/d^{r+1}) = d^3$ operations of length at least D/d^{r+1}, we conclude that an event $B_{M,i,t}$ depends on at most $2d^3 \cdot O(d^3) \cdot d^3 = O(d^9)$ other events $B_{M',i',t'}$.

Next we consider the dependencies among the congestion and contention events. It immediately follows from above that a congestion event depends on at most $d^{2+1/\epsilon} \cdot 2d^3 \cdot d^3 = 2d^{8+1/\epsilon}$ contention events, and that a contention event depends on at most $O(d^3) \cdot 2D^2$ congestion events.

Applying the results to Theorem 9.3

Now we are ready to check the requirements of Theorem 9.3. In order to transform Lemma 9.19 into a MIP, we start with defining the meaning of $\boldsymbol{x}$ and $\boldsymbol{A}$. We choose a vector $\boldsymbol{x}$ in such a way that every entry $x_{i,j}$ represents a specific delay option j of job i. $\boldsymbol{A}$ is chosen such that every row of $\boldsymbol{A}$ represents either a congestion or a contention event.

First, we consider the congestion events. For any event $A_{M,g,I}$, we intend to transform item (1) in Lemma 9.19 into a row r of $\boldsymbol{A}$ such that $(\boldsymbol{Ax})_r \geq (1+\alpha_r)y_r$ if and only if the congestion for $\hat{C}'_i(M,I)$ is too large. This is the case if, for every (i,j), we set $a_{r,(i,j)} = \ell$ if and only if the jth delay option of job i contributes an operation of size ℓ to $\hat{C}'_i(M,I)$, $\alpha_r = b_1\sqrt{d^{-(k-2)}}$, and $y_r = \hat{C}_i^{r-1}/d$. Since $a_{r,(i,j)} \leq \ell_g$ (the maximum length of operations in T_g), we obtain for $y'_r = y_r / \max_{(i,j)} a_{r,(i,j)}$ that $y'_r \geq y_r/\ell_g = d^{k-1}$. Furthermore, since

$$k - 2 \leq (1-\epsilon)(k-1) \quad \Leftrightarrow \quad k \leq 1 + \frac{1}{\epsilon}\ ,$$

we have $\alpha_r \geq (y'_r)^{-(1-\epsilon)/2}$. Thus, our ϵ can be used as it is also for Theorem 9.3. For item (2) of Lemma 9.19 we can establish a row in $\boldsymbol{A}$ in the same way as above. In this case, we obtain $y'_r \geq 1$ and $\alpha_r \geq 1$. Hence, also for item (2), $\alpha_r \geq (y'_r)^{-(1-\epsilon)/2}$.

Next we consider the contention events. For any event $B_{M,g,t}$, we transform item (3) in Lemma 9.19 into a row r of $\boldsymbol{A}$ in the same way as described above. However, now $\alpha_r = (b_2/\gamma - 1)$ and $y_r = \gamma(C/D + (\log\log D)^{1/\epsilon})$ for some constants $\gamma > 0$ and $b_2 \geq 2\gamma$. Since $a_{r,(i,j)} \leq 1$ for every (i,j) we have $y'_r \geq y^*_r \geq 1$. As $\alpha_r \geq 1$ for $b_2 \geq 2\gamma$, item (3) fulfills that $\alpha_r \geq (y'_r)^{-(1-\epsilon)/2}$. Item (4) can be transformed into a row that fulfills the same conditions.

In order to prove the other requirements of Theorem 9.3, recall that for the congestion events we obtain a probability of at most $p_C = \mathrm{e}^{-\alpha d}$ for any constant $\alpha > 0$. Furthermore, for the contention events we obtain a probability of at most $p_c = \mathrm{e}^{-\beta(\log\log D)^{1/\epsilon}}$ for any constant $\beta > 0$. Let $q_C = \mathrm{e}^{-\ln^\epsilon p_C^{-1}}$ and $z_C = q_C^\delta$. Similarly, let $q_c = \mathrm{e}^{-\ln^\epsilon p_c^{-1}}$ and $z_c = q_c^\delta$. Then

$$q_C = \mathrm{e}^{-(\alpha d)^\epsilon} \leq D^{-\alpha^\epsilon} \quad \text{and} \quad z_C \leq D^{-\delta\cdot\alpha^\epsilon}$$

and

$$q_c = \mathrm{e}^{-(\beta(\log\log D)^{1/\epsilon})^\epsilon} \leq (\log D)^{-\beta^\epsilon} \quad \text{and} \quad z_c \leq (\log D)^{-\delta\cdot\beta^\epsilon}\ .$$

Clearly, z_C and z_c can be made smaller than $1/3$, which fulfills one of the conditions of Theorem 9.3. To proceed, we need a simple claim (see Claim 7.15 for a proof).

Claim 9.20 *For any* $0 \leq x \leq 1/2$, $1 - x \geq \mathrm{e}^{-2x}$.

Let $G = (V,E)$ be the dependency graph of the events $A_{M,i,I}$ and $B_{M,i,t}$. Together with the claim, we obtain for any fixed $A_{M,i,I}$

$$\prod_{(A_{M,i,I},A_{M',i',I'})\in E} (1 - z_C)^{2/\epsilon^2} \geq \mathrm{e}^{-2(2/\epsilon^2)z_C\cdot 2d^{2+1/\epsilon}D^2} = \mathrm{e}^{-8D^{-\delta\cdot\alpha^\epsilon}d^{2+1/\epsilon}D^2/\epsilon^2} \geq \frac{1}{\mathrm{e}}$$

if α is sufficiently large. In the following, let c', c'', c''' be suitably chosen constants. For any fixed $B_{M,i,t}$

$$\prod_{(B_{M,i,t},B_{M',i',t'})} (1 - z_c)^{2/\epsilon^2} \geq \mathrm{e}^{-2(2/\epsilon^2)z_c\cdot c' d^9} = \mathrm{e}^{-4c'(\log D)^{-\delta\cdot\beta^\epsilon}d^9/\epsilon^2} \geq \frac{1}{\mathrm{e}}$$

if β is sufficiently large. Moreover, for any fixed $A_{M,i,I}$

$$\prod_{(A_{M,i,I},B_{M',i',t})} (1 - z_c)^{2/\epsilon^2} \geq \mathrm{e}^{-2(2/\epsilon^2)z_c\cdot c'' d^{8+1/\epsilon}} = \mathrm{e}^{-4c''(\log D)^{-\delta\cdot\beta^\epsilon}d^{8+1/\epsilon}/\epsilon^2} \geq \frac{1}{\mathrm{e}}$$

if β is sufficiently large. Finally, for any fixed $B_{M,i,t}$

$$\prod_{(B_{M,i,t},A_{M',i',I})} (1 - z_C)^{2/\epsilon^2} \geq \mathrm{e}^{-2(2/\epsilon^2)z_C\cdot c''' d^3 D^2} = \mathrm{e}^{-4c''' D^{-\delta\cdot\alpha^\epsilon}d^3D^2/\epsilon^2} \geq \frac{1}{\mathrm{e}}$$

if α is sufficiently large. Since $q_C \leq z_C/\mathrm{e}$ and $q_c \leq z_c/\mathrm{e}$ if α and β are sufficiently large, all requirements of Theorem 9.3 are fulfilled, which proves the lemma. □

9.4.2 Satisfiability

Many approaches have already been presented that provide good approximation algorithms for MAX SAT problems. This was pioneered by Johnson [Joh74] and Raghavan and Thompson [RT87] and further improved by Yannakakis [Yan94], Goemans and Williamson [GW94, GW95], and Asano et al. [AHOH97, AW00]. Many of these approaches use at the end a simple randomized rounding strategy. Our approach may help to improve this last step:

Consider any boolean formula in CNF. We will give an example of how to express the MAX SAT problem for this formula as a reverse form of a MIP, called *maximin integer program* ($\overline{\mathrm{MIP}}$), that is defined as follows.

Definition 9.21 *A $\overline{\mathrm{MIP}}$ has variables $\{x_{i,j} \; : \; i \in [n], \; j \in [\ell_i]\}$, for some integers ℓ_i. Let $N = \sum_{i \in [n]} \ell_i$ and let $\boldsymbol{x}$ denote the N-dimensional vector of the variables $x_{i,j}$. A $\overline{\mathrm{MIP}}$ seeks to maximize a real Y subject to:*

(1) a system of linear inequalities $\boldsymbol{Ax} \geq \boldsymbol{y}$, where $\boldsymbol{A} \in [0,1]^{m \times N}$ and $\boldsymbol{y}$ is the m-dimensional vector with the variable Y in each component,
(2) for all $i \in [n]$: $\sum_{j \in [\ell_i]} x_{i,j} = 1$, and
(3) for all i and j: $x_{i,j} \in \{0,1\}$.

Using the same dependency graph as for MIPs, the following result holds for $\overline{\mathrm{MIP}}$s.

Theorem 9.22 *We are given any $\overline{\mathrm{MIP}}$ instance I with an optimal fractional solution $\boldsymbol{x}^*$. Consider the random experiment of setting $x_{i,j}$ to 1 with probability $x^*_{i,j}$ (and in this case all other $x_{i,j'}$'s to 0). Given any vector $\boldsymbol{\alpha} \in [0,1]^m$, let*

$$p_r = \mathrm{e}^{-\alpha_r^2 y'_r/2}$$

for all $r \in [m]$, where $y'_r = y^*_r / (\max_c a_{r,c})$. ($p_r$ is a Hoeffding bound for $\Pr[(\boldsymbol{Ax})_r \leq (1-\alpha_r) y^*_r]$.) Let $0 < \epsilon < 1$ be a constant that is chosen such that $\alpha_r \geq \min[1, \; (y'_r)^{-(1-\epsilon)/2}]$ for every r. Furthermore, let $q_r = \mathrm{e}^{-\ln^\epsilon p_r^{-1}}$ and let $z_r = q_r^\delta$ be at most $1/3$ for every r, where δ is a sufficiently small positive constant. If it holds that

$$q_r \leq z_r \prod_{(r,s) \in G_I} (1 - z_s)^{2/\epsilon^2} \tag{9.5}$$

for all r, then there is an algorithm that finds a vector $\boldsymbol{x}$ in polynomial time so that $(\boldsymbol{Ax})_r \geq (1 - O(\alpha_r)) y^*_r$ for all r.

Proof. The proof of the theorem is almost identical to the proof of Theorem 9.3. Essentially, all that has to be done is to replace the Hoeffding bounds in Theorem 9.3 by Hoeffding bounds necessary for $\overline{\mathrm{MIP}}$s. Furthermore, we have to change to the rule that as long as at most $m^\epsilon_{r,k}$ many $r_{i,\ell}$'s in some set S contribute a 0 (instead of $1/\mu'$ for MIPs), then it does not affect the deviation bound for $E_{r,k}|_S$. □

An instance of MAX SAT is defined by $(\mathcal{C}, w)$, where $\mathcal{C}$ is a set of m boolean clauses such that each clause $C \in \mathcal{C}$ is a disjunction of literals with a positive weight $w(C)$. Let $X = \{x_1, \ldots, x_n\}$ be the set of boolean variables in the clauses of $\mathcal{C}$. For every i, $x_i = 1$ means x_i is true and $x_i = 0$ means x_i is false. A *literal* is a variable $x \in X$ or its negation $\bar{x} = 1 - x$.

For each x_i, we define $x_{i,0} = \bar{x}_i$ and $x_{i,1} = x_i$. We assume that no literals with the same variable appear more than once in a clause in $\mathcal{C}$. Clause j in $\mathcal{C}$ is denoted by

$$C_j = x_{i_{j,1},\ell_{j,1}} \vee x_{i_{j,2},\ell_{j,2}} \vee \ldots \vee x_{i_{j,k_j},\ell_{j,k_j}} .$$

We would like to find an optimal solution to the following integer program:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{m} w_j y_j \\ \text{subject to} \quad & \sum_{s=1}^{k_j} x_{i_{j,s},\ell_{j,s}} \geq y_j \quad \forall j \qquad (9.6) \\ & x_{i,\ell},\ y_j \in \{0,1\} \qquad (9.7) \end{aligned}$$

If we replace the conditions in (9.7) by $0 \leq x_{i,\ell}, y_j \leq 1$, then an optimal solution $\boldsymbol{x}^*$ of the LP can be found in polynomial time. In order to express this solution as a $\overline{\mathrm{MIP}}$, the variables $x_{i,\ell}$ are the same as above, and every inequality in (9.6) is represented as a row in $\boldsymbol{A}$ with the property that $(\boldsymbol{A}\boldsymbol{x}^*)_j = 1$ for every j. Obviously, the best possible Y that can be obtained for the LP relaxation of the $\overline{\mathrm{MIP}}$ is at least 1, and the value of the objective function in this case is the same as for $\boldsymbol{x}^*$.

Using our randomized rounding technique, our aim is to find integral $x_{i,\ell}$'s so that for some vector $\boldsymbol{\alpha}$, $(A\boldsymbol{x})_j \geq (1-\alpha_j)Y$ for all j. If $\boldsymbol{\alpha}$ is chosen such that the probabilities for this satisfy the conditions in Theorem 9.22, our techniques can be used to exploit non-uniform properties of the $\overline{\mathrm{MIP}}$ (and therefore the corresponding MAX SAT problem) to find good approximate solutions in polynomial time.

9.5 Conclusions

In this chapter, we presented a powerful technique to exploit non-uniform properties in MIPs in order to find good approximate solutions. We applied this technique to job shop scheduling and MAX SAT. The upper bounds we obtained for the job shop scheduling problems significantly improve previously known upper bounds. We expect our techniques to have many more applications to other types of integer programs and combinatorial optimization problems.

Bibliography

[AAF$^+$96] M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, and Z. Liu. Universal stability results for greedy contention-resolution protocols. In *Proc. of the 37th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 380–389, 1996.

[ABF93] B. Awerbuch, Y. Bartal, and A. Fiat. Competitive distributed file allocation. In *Proc. of the 25th ACM Symp. on Theory of Computing (STOC)*, pages 164–173, 1993.

[ABF96] B. Awerbuch, Y. Bartal, and A. Fiat. Distributed paging for general networks. In *Proc. of the 7th ACM Symp. on Discrete Algorithms (SODA)*, pages 574–583, 1996.

[ABLP91] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A lower bound for radio broadcast. *Journal of Computer and System Sciences*, 43(2):290–298, 1991.

[AFH$^+$97] M. Andrews, A. Fernández, M. Harchol-Balter, T. Leighton, and L. Zhang. General dynamic routing with per-packet delay guarantees of O(distance + 1 / session rate). In *Proc. of the 38th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 294–302, 1997.

[AGHP92] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–304, 1992.

[AHOH97] T. Asano, K. Hori, T. Ono, and T. Hirata. A theoretical framework of hybrid approaches to MAX SAT. In *Proc. of the 8th Symp. on Algorithms and Computation (ISAAC)*, pages 153–162, 1997.

[Alo91] N. Alon. A parallel algorithmic version of the local lemma. *Random Structures and Algorithms*, 2(4):367–387, 1991.

[Alo94] N. Alon. Probabilistic methods in coloring and decomposition problems. *Discrete Mathematics*, 127:31–46, 1994.

[AMM96] N. Alon, C. McDiarmid, and M. Molloy. Edge-disjoint cycles in regular directed graphs. *Journal on Graph Theory*, 22:231–237, 1996.

[AMR91] N. Alon, C. McDiarmid, and B. Reed. Acyclic coloring of graphs. *Random Structures and Algorithms*, 2(3):277–288, 1991.

[AS98] M. Adler and C. Scheideler. Efficient communication strategies for ad-hoc wireless networks. In *Proc. of the 10th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 259–268, 1998.

[ASE92] N. Alon, J. Spencer, and P. Erdős. *The Probabilistic Method*. John Wiley & Sons, New York, 1992.

[AW00] T. Asano and D.P. Williamson. Improved approximation algorithms for MAX SAT. In *Proc. of the 11th ACM Symp. on Discrete Algorithms (SODA)*, pages 96–105, 2000.

[Azu67] K. Azuma. Weighted sums of certain dependent random variables. *Tôkuku Math. J.*, 19:357–367, 1967.

[Bec91] J. Beck. An algorithmic approach to the Lovász local lemma. *Random Structures and Algorithms*, 2(4):343–365, 1991.

[Ben62] G. Bennet. Probability inequalities for sums of independent random variables. *Journal of the American Statistical Association*, 57:33–45, 1962.

[BFU96] A. Z. Broder, A. M. Frieze, and E. Upfal. A general approach to dynamic packet routing with bounded buffers. In *Proc. of the 37th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 390–399, 1996.

[BFU97] A. Z. Broder, A. M. Frieze, and E. Upfal. Static and dynamic path selection on expander graphs: a random walk approach. In *Proc. of the 29th ACM Symp. on Theory of Computing (STOC)*, pages 531–539, 1997.

[BGI92] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time complexity of broadcast in multi-hop radio networks: an exponential gap between determinism and randomization. *Journal of Computer and System Sciences*, 45:104–126, 1992.

[BII93] R. Bar-Yehuda, A. Israeli, and A. Itai. Multiple communication in multihop radio networks. *SIAM Journal on Computing*, 22(4):875–887, 1993.

[BKR+96] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial queueing theory. In *Proc. of the 28th ACM Symp. on Theory of Computing (STOC)*, pages 376–385, 1996.

[Boz95] S. Boztas. A robust multi-priority topology-independent transmission schedule for packet radio networks. *Information Processing Letters*, 55:291–295, 1995.

[BP97] D. Brushi and M. Del Pinto. Lower bounds for the broadcast problem in mobile radio networks. *Distributed Computing*, 10:129–135, 1997.

[BS99] P. Berenbrink and C. Scheideler. Locally efficient on-line strategies for routing packets along fixed paths. In *Proc. of the 10th ACM Symp. on Discrete Algorithms (SODA)*, pages 112–121, 1999.

[BU96] A. Broder and E. Upfal. Dynamic deflection routing on arrays. In *Proc. of the 28th ACM Symp. on Theory of Computing (STOC)*, pages 348–355, 1996.

[CF94] I. Chlamtac and A. Farago. Making transmission schedules immune to topology changes in multi-hop packet radio networks. *IEEE/ACM Transactions on Networking*, 2:23–29, 1994.

[CFM+98] R. Cole, A. Frieze, B.M. Maggs, M. Mitzenmacher, A.W. Richa, R.K. Sitaraman, and E. Upfal. On balls and bins with deletions. In *Proc. of the 2nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 1998.

[Che36] P.L. Chebychev. *Theory of Probability: Lectures given in 1879 and 1880 [Teoriya veroyatnosteĭ: Lektsii akad. P.L. Chebysheva chitannye v 1879, 1880 gg.]*. Edited by A.N. Krylov from notes by A.N. Lyapunov. Moscow–Leningrad, 1936.

[Che52] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–509, 1952.

[Che94] K.-C. Chen. Medium access control of wireless LANs for mobile computing. *IEEE Network*, 8(5):50–63, 1994.

[CK85] I. Chlamtac and S. Kutten. On broadcasting in radio networks – problem analysis and protocol design. *IEEE Transactions on Communications*, 33(12), 1985.

[CMM+98] R. Cole, B.M. Maggs, F. Meyer auf der Heide, M. Mitzenmacher, A.W. Richa, K. Schröder, R.K. Sitaraman, and B. Vöcking. Randomized protocols for low-congestion circuit routing in multistage interconnection networks. In *Proc. of the 29th ACM Symp. on Theory of Computing (STOC)*, pages 378–388, 1998.

[CMS96] R. J. Cole, B. M. Maggs, and R. K. Sitaraman. On the benefit of supporting virtual channels in wormhole routers. In *Proc. of the 8th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 131–141, 1996.

[CMS97] R. Cole, B. Maggs, and R. Sitaraman. Reconfiguring arrays with random faults. *SIAM Journal on Computing*, 26(6):1581–1611, 1997.

[CMSV96] R. Cypher, F. Meyer auf der Heide, C. Scheideler, and B. Vöcking. Universal algorithms for store-and-forward and wormhole routing. In *Proc. of the 28th ACM Symp. on Theory of Computing (STOC)*, pages 356–365, 1996.

[CP87] I. Chlamtac and S. Pinter. A distributed nodes organization algorithm for channel access in a multi-hop dynamic radio network. *IEEE Transactions on Computers*, 36:728–737, 1987.

[CS89] I. Cidon and M. Sidi. Distributed assignment algorithms for multihop packet radio networks. *IEEE Transactions on Computers*, 38(10):1353–1361, 1989.

[CS00a] A. Czumaj and C. Scheideler. Coloring non-uniform hypergraphs: a new algorithmic approach to the general Lovász local lemma. In *Proc. of the 11th ACM Symp. on Discrete Algorithms (SODA)*, pages 30–39, 2000.

[CS00b] A. Czumaj and C. Scheideler. A new algorithmic approach to the general Lovász local lemma with applications to scheduling and satisfiability problems. In *Proc. of the 32nd ACM Symp. on Theory of Computing (STOC)*, 2000.

[CW91] I. Chlamtac and O. Weinstein. The wave expansion approach to broadcasting in multihop radio networks. *IEEE Transactions on Communications*, 39(3):426–433, 1991.

[DKKP95] S. Dolev, E. Kranakis, D. Krizanc, and D. Peleg. Bubbles: adaptive routing scheme for high-speed dynamic networks. In *Proc. of the 27th ACM Symp. on Theory of Computing (STOC)*, pages 528–537, 1995.

[DP98] D. Dubhashi and A. Panconesi. Concentration of measure for the analysis of randomized algorithms. Unpublished manuscript, accessible via http://www.cs.unibo.it/∼pancones/papers.html, October 20 1998.

[DR94] D. Dubhashi and D. Ranjan. Come correlation inequalities for probabilistic analysis of algorithms. Technical report, MPI-I-94-143, Max Planck Institut für Informatik, Saarbrücken, Germany, 1994.

[DSB97] B. Das, R. Sivakuma, and V. Bharghavan. Routing in ad-hoc networks using a virtual backbone. In *Proc. of the IEEE IC3N*, 1997.

[EGL+98] G. Even, O. Goldreich, M. Luby, N. Nisan, and B. Veličković. Efficient approximations of product distributions. *Random Structures and Algorithms*, 13(1):1–16, 1998.

[EL75] P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In A. Hajnal, R. Rado, and V.T. Sós, editors, *Infinite and Finite Sets (to Paul Erdős on his 60th birthday)*, pages 609–627. North-Holland, Amsterdam, 1975.

[Erd47] P. Erdős. Some remarks on the theory of graphs. *Bulletin of the American Mathematical Society*, 53:292–294, 1947.

[Erd59] P. Erdős. Graph theory and probability. *Canadian Journal of Mathematics*, 11:34–38, 1959.

[Erd61] P. Erdős. Graph theory and probability II. *Canadian Journal of Mathematics*, 13:346–352, 1961.

[ES91] P. Erdős and J. Spencer. Lopsided Lovász local lemma and latin transversals. *Discrete Applied Mathematics*, 30:151–154, 1991.

[FGK71] C.M. Fortuin, J. Ginibre, and P.N. Kasteleyn. Correlation inequalities for partially ordered sets. *Communications of Mathematical Physics*, 22:89–103, 1971.

[FK86] Z. Füredi and J. Kahn. On the dimensions of ordered sets of bounded degree. *Order*, 3:15–20, 1986.

[FK96] U. Feige and J. Kilian. Zero knowledge and the chromatic number. In *Proc. of the 11th IEEE Conf. on Computational Complexity*, pages 278–287, 1996.

[FS98] U. Feige and C. Scheideler. Improved bounds for acyclic job shop scheduling. In *Proc. of the 30th ACM Symp. on Theory of Computing (STOC)*, pages 624–633, 1998.

[Gal85] R. Gallager. A perspective on multiaccess channels. *IEEE Transactions on Information Theory*, 31:124–142, 1985.

[GGL95] R. Graham, M. Grötschel, and L. Lovász, editors. *Handbook of Combinatorics*. Elsevier Science, Amsterdam, 1995.

[Gib96] J.D. Gibson, editor. *The Mobile Communications Handbook*. CRC Press, 1996.

[GK68] B.V. Gnedenko and A.N. Kolmogorov. *Limit Distributions for Sums of Independent Random Variables*. Addison-Wesley, Reading, 1968.

[Gle75] L. Gleser. On the distribution of the number of successes in independent trials. *Ann. Probab.*, 3:182–188, 1975.

[GM95] I. Gaber and Y. Mansour. Broadcast in radio networks. In *Proc. of the 6th ACM Symp. on Discrete Algorithms (SODA)*, pages 577–582, 1995.

[GPSS97] L. A. Goldberg, M. Paterson, A. Srinivasan, and E. Sweedyk. Better approximation guarantees for job-shop scheduling. In *Proc. of the 8th ACM Symp. on Discrete Algorithms (SODA)*, pages 599–608, 1997.

[GW94] M.X. Goemans and D.P. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Computing*, 7:656–666, 1994.

[GW95] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[HMR97] H. Hind, M. Molloy, and B. Reed. Colouring a graph frugally. *Combinatorica*, 17(4):469–482, 1997.

[HMRR98] M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed, editors. *Probabilistic Methods for Algorithmic Discrete Mathematics*. Springer Verlag, Berlin, 1998.

[Hoa62] C.A.R. Hoare. Quicksort. *Computer Journal*, 5:10–15, 1962.

[Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, 58:13–30, 1963.

[Hof87] M. Hofri. *Probabilistic Analysis of Algorithms*. Springer Verlag, Heidelberg, 1987.

[HR90] T. Hagerup and C. Rüb. A guided tour of Chernoff bounds. *Information Processing Letters*, 33:305–308, 1989/90.

[HW95] M. Harchol-Balter and D. Wolfe. Bounding delays in packet-routing networks. In *Proc. of the 27th ACM Symp. on Theory of Computing (STOC)*, pages 248–257, 1995.

[Hwa95] F.K. Hwang. The time complexity of deterministic broadcast radio networks. *Discrete Applied Mathematics*, 60:219–222, 1995.

[IE96] T. Imielinski and H. Korth (Eds.). *Mobile Computing*. Kluwer Academic Publishers, 1996.

[IM74] K. Ito and H.P. McKean. *Diffusion Processes and their Sample Paths*. Springer Verlag, Heidelberg, 1974.

[Jac96] J.M. Jacobsmeyer. Congestion relief on power-controlled CDMA networks. *IEEE Journal on Selected Areas in Communications*, 18(9):1758–1761, 1996.

[JK77] N. L. Johnson and S. Kotz. *Urn Models and Their Application*. John Wiley & Sons, New York, 1977.

[JK82] N.L. Johnson and S. Kotz. *Encyclopedia of Statistical Sciences*. John Wiley, New York, 1982.

[JM96] D.B. Johnson and D.A. Maltz. Protocols for adaptive wireless and mobile networking. *IEEE Personal Comm.*, 3(1), 1996.

[Joh74] D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[JP83] K. Joag-Dev and F. Proschan. Negative association of random variables, with applications. *The Annals of Statistics*, 11(1):286–295, 1983.

[Kar91] R.M. Karp. An introduction to randomized algorithms. *Discrete Applied Mathematics*, 34:165–201, 1991.

[KKKP97] L.M. Kirousis, E. Kranakis, D. Krizanc, and A. Pelc. Power consumption in packet radio networks. In *Proc. of the 14th Symp. on Theoretical Aspects of Computer Science (STACS)*, 1997.

[KKL$^+$90] C. Kaklamanis, A.R. Karlin, F.T. Leighton, V. Milenkovic, P. Raghavan, S.B. Rao, C. Thomborson, and A. Tsantilas. Asymptotically tight bounds for computing with faulty arrays of processors. In *Proc. of the 31st IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 285–296, 1990.

[KL95] N. Kahale and T. Leighton. Greedy dynamic routing on arrays. In *Proc. of the 6th ACM Symp. on Discrete Algorithms (SODA)*, pages 558–566, 1995.

[KLM92] R. Karp, M. Luby, and F. Meyer auf der Heide. Efficient PRAM simulation on a distributed memory machine. In *Proc. of the 24th ACM Symp. on Theory of Computing (STOC)*, pages 318–326, 1992.

[KM93] E. Kushilevitz and Y. Mansour. An $\omega(d\log(n/d))$ lower bound for broadcast in radio networks. In *Proc. of the 12th IEEE Symp. on Principles of Distributed Computing (PODC)*, 1993.

[Knu69] D. Knuth. *The Art of Computer Programming, Vol. I.* Addison Wesley, London, 1969.

[KR95] S. Singh K. Ravishankar. Asymptotically optimal gossiping in radio networks. *Discrete Applied Mathematics*, 61:61–82, 1995.

[LMR88] F. T. Leighton, B. M. Maggs, and S. B. Rao. Universal packet routing algorithms. In *Proc. of the 29th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 256–271, 1988.

[LMR94] F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and job-shop scheduling in O(congestion + dilation) steps. *Combinatorica*, 14(2):167–186, 1994.

[LMR99] F. T. Leighton, B. M. Maggs, and A. W. Richa. Fast algorithms for finding o(congestion + dilation) packet routing schedules. *Combinatorica*, 19(2):1–27, 1999.

[LMRR94] F. T. Leighton, B. M. Maggs, A. G. Ranade, and S. B. Rao. Randomized routing and sorting in fixed-connection networks. *Journal of Algorithms*, 17:157–205, 1994.

[Loé77] M. Loéve. *Probability theory*. Springer Verlag, New York, 1977.

[LRS99] T. Leighton, S. Rao, and A. Srinivasan. New algorithmic aspects of the Local Lemma with applications to routing and partitioning. In *Proc. of the 10th ACM Symp. on Discrete Algorithms (SODA)*, pages 643–652, 1999.

[LRWY96] C. Lund, R. Reingold, J. Westbrook, and D. Yan. On-line distributed data management. In *Proc. of the 2nd European Symposium on Algorithms (ESA)*, 1996.

[Lu99] C.-J. Lu. A deterministic approximation algorithm for a minmax integer programming problem. In *Proc. of the 10th ACM Symp. on Discrete Algorithms (SODA)*, pages 663–668, 1999.

[Mar13] A.A. Markov. *Calculus of Probabilities [Ischislenie veroyatnosteĭ]*. 3rd ed. St. Petersburg, 1913.

[McD89] C. McDiarmid. On the method of bounded differences. In J. Siemons, editor, *Surveys in Combinatorics*. London Mathematical Society Lecture Note Series 141, Cambridge University Press, 1989.

[McD98] McDiarmid. Concentration. In M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed, editors, *Probabilistic Methods for Algorithmic Discrete Mathematics*, pages 195–247. Springer Verlag, Berlin, 1998.

[Mik93] V.G. Mikhailov. On a refinement of the Central Limit Theorem for sums of independent random indicators. *Theory of Probability and Its Applications*, 38(3):479–489, 1993.

[Mit94] M. Mitzenmacher. Bounds on the greedy routing algorithm for array networks. In *Proc. of the 6th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 346–353, 1994.

[MJP95] A. Myles, D.B. Johnson, and C.E. Perkins. A mobile host protocol supporting route optimization and authentication. *IEEE Journal on Selected Areas in Communications*, 13(5):839–849, 1995.

[MMVW97] B. Maggs, F. Meyer auf der Heide, B. Vöcking, and M. Westermann. Exploiting locality for data management in systems of limited bandwidth. In *Proc. of the 38th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 284–293, 1997.

[Mol98] M. Molloy. The probabilistic method. In M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed, editors, *Probabilistic Methods for Algorithmic Discrete Mathematics*, pages 1–35. Springer Verlag, Berlin, 1998.

[MP97] M. Merkle and L. Petrovic. Inequalities for sums of independent geometrical random variables. *Aequationes Mathematicae*, 54:173–180, 1997.

[MR95] R. Motwani and P. Rahgavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[MR98] M. Molloy and B. Reed. Further algorithmic aspects of the local lemma. In *Proc. of the 30th ACM Symp. on Theory of Computing (STOC)*, pages 524–529, 1998.

[MSV85] F. Maffioli, M.G. Speranza, and C. Vercellis. Randomized algorithms. In M. O'hEigertaigh, J.K. Lenstra, and A.H.G. Rinooy Kan, editors, *Combinatorial Optimization: Annotated Bibliographies*, pages 89–105. John Wiley, New York, 1985.

[MV95] F. Meyer auf der Heide and B. Vöcking. A packet routing protocol for arbitrary networks. In *Proc. of the 12th Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 291–302, 1995.

[MV99] F. Meyer auf der Heide and B. Vöcking. Shortest paths routing in arbitrary networks. *Journal of Algorithms*, 31(1):105–131, 1999.

[Nag79] S.V. Nagaev. Large deviations of sums of independent random variables. *The Annals of Probability*, 7:745–789, 1979.

[Nav97] P. Naveau. Comparison between the Chernoff and factorial moment bounds for discrete random variables. *The American Statistician*, 51(1):40–41, 1997.

[NN93] J. Naor and J. Naor. Small-bias probability spaces: efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993.

[OR97] R. Ostrovsky and Y. Rabani. Universal O(congestion + dilation + $\log^{1+\epsilon} n$) local control packet switching algorithms. In *Proc. of the 29th ACM Symp. on Theory of Computing (STOC)*, pages 644–653, 1997.

[Pet95] V.V. Petrov. *Limit Theorems of Probability Theory*. Clarendon Press, Oxford, 1995.

[Pir91] P. Piret. On the connectivity of radio networks. *IEEE Transactions on Information Theory*, 37(5):1490–1492, 1991.

[PN95] T. K. Philips and R. Nelson. The moment bound is tighter than Chernoff's bound for positive tail estimates. *American Statistical Association*, 49(2):175–178, 1995.

[Pre85] E.L. Presman. Approximation in variation of the distribution of a sum of independent Bernoulli variables with a Poisson law. *Theory of Probability and Its Applications*, 30(2):417–422, 1985.

[PS92] A. Panconesi and A. Srinivasan. Fast randomized algorithms for distributed edge coloring. In *Proc. of the 11th IEEE Symp. on Principles of Distributed Computing (PODC)*, pages 251–262, 1992.

[PSK85] M.J. Post, P.E. Sarachik, and A.S. Kershenbaum. A distributed evolutionary algorithm for reorganizing network communications. In *Proc. of the IEEE MILCOM*, 1985.

[Rag88] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences*, 37:130–143, 1988.

[Rag89] P. Raghavan. Robust algorithms for packet routing in a mesh. In *Proc. of the 1st ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 344–350, 1989.

[Ran91] A. G. Ranade. How to emulate shared memory. *Journal of Computer and System Sciences*, 42:307–326, 1991.

[Ree98] B. Reed. ω, Δ, and χ. *Journal of Graph Theory*, 27(4):177–212, 1998.

[Rob72] L.G. Roberts. Aloha packet system with and without slots and capture. Technical report, ASS Notes 8, Advanced Research Projects Agency, Network Information Center, Stanford Research Institute, Stanford, CA, 1972.

[RS98] J. Radhakrishnan and A. Srinivasan. Improved bounds and algorithms for hypergraph two-coloring. In *Proc. of the 39th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 684–693, 1998.

[RT87] P. Raghavan and C.D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.

[RT96] Y. Rabani and E. Tardos. Distributed packet switching in arbitrary networks. In *Proc. of the 28th ACM Symp. on Theory of Computing (STOC)*, pages 366–375, 1996.

[Sch93] C. Scheideler. Analysis of shared memory simulations that use several hash functions (in German). Master's thesis, Dept. of Mathematics and Computer Science, Paderborn University, Germany, June 1993.

[Sch98] C. Scheideler. *Universal Routing Strategies for Interconnection Networks*. Springer Verlag, Heidelberg, 1998.

[SH96] A. Sen and M. Huson. A new model for scheduling packet radio networks. In *Proc. of the 15th IEEE INFOCOM*, pages 1116–1124, 1996.

[She85] Shearer. On a problem of Spencer. *Combinatorica*, 5:241–245, 1985.

[Shi96] A.N. Shiryaev. *Probability*. Springer Verlag, New York, 1996.

[Sie89] J. Siemons, editor. *Surveys in Combinatorics*. London Mathematical Society Lecture Note Series 141, Cambridge University Press, 1989.

[Spe95] J. Spencer. Probabilistic methods. In R. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of Combinatorics*, chapter 33. Elsevier Science, Amsterdam, 1995.

[Sri96] A. Srinivasan. An extension of the Lovász local lemma, and its applications to integer programming. In *Proc. of the 7th ACM Symp. on Discrete Algorithms (SODA)*, pages 6–15, 1996.

[SSS95] J. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics*, 8(2):223–250, 1995.

[SSW94] D. Shmoys, C. Stein, and J. Wein. Improved approximation algorithms for shop scheduling problems. *SIAM Journal on Computing*, 23(3):617–632, 1994.

[ST91] G. D. Stamoulis and J. N. Tsitsiklis. The efficiency of greedy routing in hypercubes and butterflies. In *Proc. of the 3rd ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 248–259, 1991.

[SV96] C. Scheideler and B. Vöcking. Universal continuous routing strategies. In *Proc. of the 8th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 142–151, 1996.

[SV99] C. Scheideler and B. Vöcking. From static to dynamic routing: efficient transformations of store-and-forward protocols. In *Proc. of the 31st ACM Symp. on Theory of Computing (STOC)*, pages 215–224, 1999.

[Sze43] T. Szele. Kombinatorikai vizsgálatok az irányitott teljes gráffal kapcsolatban. *Mat. Fiz. Lapok*, 50:233–256, 1943.

[Tal95a] M. Talagrand. Concentration of measure and isoperimetric inequalities in product spaces. *Institut Des Hautes Études Scientifiques, Publications Mathématiques*, 81:73–205, 1995.

[Tal95b] M. Talagrand. The missing factor in Hoeffding's inequalities. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, 31(4):689–702, 1995.

[Tal96] M. Talagrand. A new look at independence. *The Annals of Probability*, 24(1):1–34, 1996.

[Tur34] P. Turán. On a theorem of Hardy and Ramanujan. *Journal of the London Mathematical Society*, 9:274–276, 1934.

[UY96] S. Ulukus and R.D. Yates. Stochastic power control for cellular radio systems. Technical report, Dept. of Electrical and Computer Engineering, Rutgers University, NJ, 1996.

[Val82] L.G. Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing*, 11(2):350–361, 1982.

[Wei87] O. Weinstein. The wave expansion approach to broadcasting in multihop networks. Master's thesis, Computer Science Dept., Technion, Haifa, Israel, 1987.

[Wel83] D.J.A. Welsh. Randomised algorithms. *Discrete Applied Mathematics*, 5:133–145, 1983.

[WHH+96] D. Williamson, L. Hall, J. Hoogeveen, C. Hurkens, J. Lenstra, S. Sevastjanov, and D. Shmoys. Short shop schedules. *Operations Research*, 1996.

[Yan94] M. Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms*, 17:475–502, 1994.