3 Network Flows II – Single-Commodity Flows

In the previous section we only looked at how to find least-cost paths for source-destination pairs. However, it is not too difficult to see that least-cost paths can cause a high congestion, i.e. many paths may go across the same edge. Thus, it may be better to use some kind of coordination among the paths so that no edge gets overloaded. We start investigating this approach by just looking at the case that all information has to be sent to a single sink, i.e. we only have a single-commodity flow.

A *flow* from a source s to a sink t is a real-valued function $f : V \times V \to \mathbb{R}^+$ that satisfies

- **Capacity constraints:** For all $u, v \in V$, $f(u, v) \leq c(u, v)$.
- **Flow conservation:** For all $u \in V \setminus \{s, t\}$, $\sum_{v \in V} f(u, v) = 0$.
- **Skew symmetry:** For all $u, v \in V$, $f(u, v) = -f(v, u)$.

The *value* of a flow f is defined as $|f| = \sum_{u \in V} f(s, u)$. There are two classical single-commodity flow problems: the maximum-flow problem and the mincost flow problem.

In the maximum-flow problem we are given a network $G = (V, E)$ with edge capacities $c : E \to \mathbb{R}^+$ and a source-destination pair (s, t), and the problem is to find a flow of maximum value from s to t.

In the minimum-cost flow problem we are given a network $G = (V, E)$ with edge capacities determined by $c : E \to \mathbb{R}^+$ and edge weights determined by $w : E \to \mathbb{R}^+$. Furthermore, we have a source s, a sink t, and a parameter φ, and the problem is to find a flow f with $|f| = \varphi$ from s to t that minimizes

$$\sum_{e:\, f(e)>0} f(e) \cdot w(e) \ .$$

Before we present distributed algorithms for these flow problems, we investigate some basic properties of single-commodity flows.

3.1 The max-flow min-cut theorem

An *s-t cut* is a partition (S, T) of the graph G with $S \cup T = V$, $s \in S$, $t \in T$, and $S \cap T = \emptyset$. The *capacity* of a cut (S, T) is defined as

$$c(S, T) = \sum_{u \in S,\, v \in T} c(u, v)$$

and the flow over a cut (S, T) is defined as

$$f(S, T) = \sum_{u \in S,\, v \in T} f(u, v)$$

See Figure 1 for an example of a flow and a cut. We start with an important observation about flows and cuts.

Lemma 3.1 *For any s-t cut (S, T) and any s-t flow f, $|f| = f(S, T)$.*

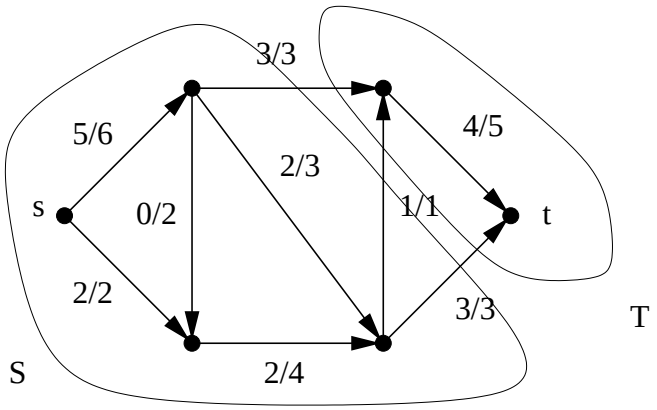

Figure 1: An example of a flow and a cut. In the 'x/y' expressions, y represents the capacity of an edge and x the flow sent along it.

Proof. Consider any s-t cut (S', T') and suppose that for this cut, $|f| = f(S', T')$. Now, take any node $v \in T' \setminus \{t\}$, and consider the cut (S'', T'') with $S'' = S' \cup \{v\}$. From the flow conservation principle we know that $f(\{v\}, S') + f(\{v\}, T') = 0$. Hence, it follows from the skew symmetry that

$$\begin{aligned} f(S'', T'') &= f(S', T') - f(S', \{v\}) + f(\{v\}, T') \\ &= f(S', T') + f(\{v\}, S') + f(\{v\}, T') = f(S', T') \ . \end{aligned}$$

Since by the definition of a flow, $f(\{s\}, V \setminus \{s\}) = |f|$, and every s-t cut (S, T) can be constructed from $(\{s\}, V \setminus \{s\})$ by successively adding a node, the lemma follows. □

The lemma immediately implies the following result.

Corollary 3.2 *The maximum value of an s-t flow can be at most the minimum capacity of an s-t cut, i.e.*

$$\max_f |f| \leq \min_{(S,T)} c(S, T) \ .$$

Is it also possible to show that $\max_f |f| \geq \min_{(S,T)} c(S, T)$? To answer this question, we need the concept of residual graphs.

Given a network $G = (V, E)$ with flow f, the *residual network* $G_f = (V, E_f)$ is a directed network containing all edges $e \in E$ with positive *residual capacities* $c_f(e) = c(e) - f(e)$. An *augmenting path* is a directed s-t path along edges in G_f. The residual capacity of an augmenting path p is defined as

$$c_f(p) = \min_{e \in p} c_f(e) \ .$$

Lemma 3.3 *Given G_f and an augmenting path with residual capacity c, one can augment the flow value $|f|$ by c without violating the flow constraints.*

Proof. Suppose that there is an augmenting path p with residual capacity c. If we define the flow f_p with $f_p(e) = c$ for all $e \in p$, then this is a valid flow in G_f. Consider now the flow f' with flow values $f'(e) = f(e) + f_p(e)$ for all edges e. Since the superposition of two valid flows preserve the

flow conservation principle and skew symmetry, we only have to worry about the capacity constraints. However, since for every edge $e \in p$ we know that $c_f(e) = c(e) - f(e) \geq c$, it holds for each of these edges that $f(e) + f_p(e) \leq c(e)$. Hence, also the capacity constraints are kept, which completes the proof. □

Now we can prove the following result:

Lemma 3.4 *For any network G,*

$$\max_f |f| \geq \min_{(S,T)} c(S,T) \ .$$

Proof. Suppose that f is a flow of maximum possible value. Let S_f be the set of all nodes in G_f reachable from s. If S_f includes t, there must be an augmenting path in G_f and therefore $|f|$ cannot be maximal. Hence, S_f cannot include t. However, in this case it must hold for all $u \in S_f$ and $v \in V \setminus S_f$ that $f(u,v) \geq c(u,v)$, and therefore $f(S_f, V \setminus S_f) \geq c(S_f, V \setminus S_f)$. Since $f(S_f, V \setminus S_f) = |f|$, the lemma follows. □

Combining Corollary 3.2 and Lemma 3.4, we get:

Theorem 3.5 (Max-flow min-cut theorem) *For any network G, the maximum s-t flow is equal to the minimum s-t cut, i.e.*

$$\max_f |f| = \min_{(S,T)} c(S,T) \ .$$

3.2 The preflow algorithm

Next we present an algorithm, called *preflow algorithm* (see e.g. [2]), that finds a maximum flow in a distributed way. Unlike other flow algorithms, the preflow algorithm does not try to find augmenting paths but first pushes a maximum amount of flow out from the source and then adjusts the flow by letting as much of it as possible to the sink and returning any excess flow to the source.

In a *preflow*, the flow conservation principle is modified in the following way:

- **Preflow conservation:** For all $v \in V \setminus \{s,t\}$, $e(v) = \sum_{u \in V} f(u,v) \geq 0$.

Here, $e(v)$ is the *excess* at node v. Note that $e(t) \geq 0$ since no flow leaves t. $e(s) \leq 0$, but s is the only node with negative excess. The preflow algorithm assigns a *height* $h(v)$ to each node v such that the following conditions are satisfied:

1. $h(s) = n$
2. $h(t) = 0$
3. $h(u) \leq h(v) + 1$ for all $(u,v) \in E_f$

The goal is to push flow towards the sink. We determine the direction of the *push* operation from the heights of the neighboring nodes: flows can only be pushed from higher nodes to lower nodes. As we will see in a moment, the algorithm proceeds as long as there is a node with excess. Initially, we have the following situation:

- $h(u) = 0$ for all $u \in V \setminus \{s\}$ and $h(s) = n$,

- $f(u,v) = f(v,u) = 0$ for all $(u,v) \in E$ with $u, v \neq s$, and
- $f(s,u) = c(s,u)$ and $f(u,s) = -f(s,u)$ for all $(s,u) \in E$.

Afterwards, the preflow algorithm proceeds by repeating the steps shown in Figure 2.

Preflow Algorithm:
At each node u:
  if $e(u) > 0$ and there is a neighbor v with $c_f(u,v) > 0$ and $h(u) = h(v) + 1$
    Push: $p_f(u,v) = \min\{e(u), c_f(u,v)\}$
      $f(u,v) = f(u,v) + p_f(u,v)$
      $f(v,u) = f(v,u) - p_f(u,v)$
  else if $e(u) > 0$ and $h(u) \leq h(v)$ for all $(u,v) \in E_f$
    Lift: $h(u) = h(u) + 1$

Figure 2: The preflow algorithm for the maximum-flow problem.

Next we show that the algorithm indeed finds a maximum flow. The following lemma is crucial for the rest of the analysis.

Lemma 3.6 *During the execution of the preflow algorithm, $h(u) \leq h(v) + 1$ for all $(u,v) \in E_f$.*

Proof. We prove the lemma by complete induction over the number of rounds of the preflow algorithm. At the beginning, the lemma is certainly true. So suppose that it is true up to some round t. If there are no further push or lift operations afterwards, we are done. Consider any node u with a push or lift operation in round $t+1$.

Suppose that u performs a push operation. Then there is a neighbor v with $(u,v) \in E_f$ and $h(u) = h(v) + 1$. Whatever operation v performs, it holds at the beginning of step $t+1$ that $h(v) \in \{h(u) - 1, h(u)\}$. Hence, for both (u,v) and (v,u) the height condition will not be violated.

So suppose that u performs a lift operation. This does not endanger the condition $h(u) \leq h(v) + 1$ or $h(v) \leq h(u) + 1$ for any neighbor v of u performing a push operation, because we already considered push operations in the first case. So let v be any neighbor of u that either does nothing or performs a lift operation. If $(u,v) \notin E_f$, there is nothing to show. So suppose that $(u,v) \in E_f$. In this case, we know that $h(u) \leq h(v)$, and therefore $h(u) \leq h(v) + 1$ after u's lift operation, whatever v does. Hence, the lemma is true. □

Lemma 3.7 *For any node $u \in V \setminus \{s,t\}$ it holds that if $e(u) > 0$, then either a push or a lift operation applies to u.*

Proof. Consider any node u with $e(u) > 0$. If a push operation can be performed, we are fine. So suppose that there is no $(u,v) \in E_f$ with $h(u) = h(v) + 1$. Then it follows from the previous lemma that $h(u) \leq h(v)$ for all $(u,v) \in E_f$, and we can therefore perform a lift operation. □

Lemma 3.8 *There is never a path from s to t in G_f.*

Proof. By the property of the height function, $h(u) \leq h(v) + 1$ for every $(u, v) \in E_f$. If there is a path p from s to t, then it spans at most $n - 1$ edges. Summing up the height inequalities on those edges, we have $h(s) \leq h(t) + n - 1 = n - 1$, but $h(s) = n$, leading to a contradiction. □

Combining the lemmata above, we get:

Theorem 3.9 *The preflow algorithm terminates with a maximum flow on G.*

Proof. When the preflow algorithm terminates, $e(u) = 0$ for all $u \in V \setminus \{s, t\}$, and hence we have a valid flow. This is a maximum flow because there is no path from s to t in G_f. □

Next we argue that the algorithm is guaranteed to terminate. Consider the lift operation. It is clear that only nodes in $V \setminus \{s, t\}$ are lifted. Hence, it follows from Lemma 3.6 that the maximum height of a node can be at most $2n - 1$, and therefore the lift operation can be applied at most $2n - 1$ times to any node.

Consider now a saturating push operation. The necessary condition for a saturating push along edge (u, v) is $h(u) = h(v) + 1$. In order for this operation to be performed again, there must be a push from v to u, which can only happen when $h(v)$ has been increased by at least 2. Similarly, $h(u)$ must increase by at least 2 in order to perform another push from u to v. Hence, there can be at most $2n - 1$ saturating pushes along every edge.

It remains to consider the non-saturating pushes. To handle this case, we define a potential function $\Phi = \sum_{v \in X} h(v)$ where X is the set of overflowing nodes. Initially, $\Phi = 0$. Consider the operations to be executed one by one. There are three possible ways that Φ can be changed:

1. **Lift operation:** In this case, the set X remains the same, and for each node lifted, $h(v)$ is increased by (at most) 1.

2. **Saturating push:** In this case, each node keeps the same height. But after a saturating push from u to v, v can possibly be added to X, which incurs an increase of at most $2n - 1$ in Φ.

3. **Non-saturating push:** Each node remains at the same height. But after a non-saturating push from u to v, u leaves X and v enters X. Since $h(u) = h(v) + 1$, this decreases Φ by (at least) 1.

Because the number of lift and saturating push operations is bounded, there is an upper bound on the total value they can contribute to Φ. Since it must hold that $\Phi \geq 0$ and a non-saturating push can only decrease Φ, this means that there must be a limited number of non-saturating pushes. Hence, the preflow algorithm terminates.

3.3 The T-balancing algorithm

For simplicity, we assume in this section that $c(e) = 1$ and $w(e) = 1$ for every edge e.

Given a minimum-cost flow problem, we consider the following situation. For every node v, let $h(v)$ be the (absolute) amount of flow stored in it, and let $\bar{h}(v) = h(v)/d$ be the relative amount of flow stored in it, where d is the (maximum) degree of G. Initially, there is no flow at any node, i.e. each node v has a height $h(v)$ of 0. Afterwards, a flow of φ is injected at node s in each time step, i.e. its height $h(s)$ is increased by φ in each step. The balancing algorithm shown in Figure 3 is used to

T-**Balancing Algorithm**:

At each node u:

for each edge (u, v) **do**
if $\bar{h}(u) - \bar{h}(v) > T$
send a flow of $\min\{\bar{h}(u) - \bar{h}(v) - T, 1\}$ from u to v

Figure 3: The T-balancing algorithm for the minimum-cost flow problem.

send the injected flow towards the destination t. (The parameter T used in it will be specified later.) Any flow that reaches t will be absorbed, i.e. $h(t) = 0$ at any time.

We start with a statement that the T-balancing algorithm is *stable*, i.e. if the injected flow is feasible, it will find flow paths to the destination so that the height of every node is bounded at any time.

Theorem 3.10 *For any $T \geq 1$ it holds: If φ is feasible, then the T-balancing algorithm ensures that $h(v) \leq (\varphi + T - 1)n$ for every node v at any point in time.*

The proof is quite complicated and therefore left out here. For details, see [1]. The next theorem shows that the algorithm does not only guarantee bounded heights but that it will approach a *fixpoint*, i.e. a point in which the amount of flow at every node remains unchanged when applying the balancing algorithm to it.

Theorem 3.11 *For any static injection pattern and any $T \geq 1$, the distribution of the flow will converge towards a fixpoint.*

Proof. Let $\bar{h}_{v,t}$ denote the normalized height of node v at time step t and $\delta_{v,t} = \bar{h}_{v,t+1} - \bar{h}_{v,t}$. We will show that if the system starts with $\bar{h}_{v,0} = 0$ for every node v, then $\delta_{v,t} \geq 0$ for all v and t. Since this implies that the (normalized) heights of the nodes can only grow, and since we know from Theorem 3.10 that the total amount of flow in the system must be bounded, this implies that the system must converge towards a state with $\delta_{v,t} = 0$ for all v, i.e. a fixpoint.

Lemma 3.12 *If $\bar{h}_{v,0} = 0$ for every node v, then $\delta_{v,t} \geq 0$ for all $t \geq 0$ and all $v \in V$.*

Proof. To simplify the proof, we will view the injected flow as flow along edges coming from imaginary nodes v for which w.l.o.g. we have $\delta_{v,t} \geq 0$ for all $t \geq 0$.

At the beginning, the lemma is obviously true for the empty system. Now suppose that we already showed for some time step t that $\delta_{v,t} \geq 0$ for all $v \in V$. Then we will show that it also holds for step $t + 1$.

Consider an arbitrary node v. We know that $\bar{h}_{v,t+1} = \bar{h}_{v,t} + \delta_{v,t}$ where $\delta_{v,t} \geq 0$. Since also $\bar{h}_{w,t+1} \geq \bar{h}_{w,t}$ for all other nodes w, we get

$$\bar{h}_{v,t+2} = \bar{h}_{v,t+1} + \tfrac{1}{d}\left(\sum_{w:\ \bar{h}_{w,t+1} > \bar{h}_{v,t+1}+T} \min[1,\ \bar{h}_{w,t+1} - \bar{h}_{v,t+1} - T] - \sum_{w:\ \bar{h}_{w,t+1} < \bar{h}_{v,t+1}-T} \min[1,\ \bar{h}_{v,t+1} - \bar{h}_{w,t+1} - T]\right)$$

$$
\begin{aligned}
&\geq \bar{h}_{v,t+1} + \tfrac{1}{d}\left(\sum_{w:\,\bar{h}_{w,t}>\bar{h}_{v,t+1}+T} \min[1,\ \bar{h}_{w,t}-\bar{h}_{v,t+1}-T] - \sum_{w:\,\bar{h}_{w,t}<\bar{h}_{v,t+1}-T} \min[1,\ \bar{h}_{v,t+1}-\bar{h}_{w,t}-T]\right)\\
&\geq \bar{h}_{v,t+1} + \tfrac{1}{d}\left(\sum_{w:\,\bar{h}_{w,t}>\bar{h}_{v,t}+T} \min[1,\ \bar{h}_{w,t}-\bar{h}_{v,t+1}-T] - \sum_{w:\,\bar{h}_{w,t}<\bar{h}_{v,t+1}-T} \min[1,\ \bar{h}_{v,t+1}-\bar{h}_{w,t}-T]\right)\\
&= \bar{h}_{v,t+1} + \tfrac{1}{d}\left(\sum_{w:\,\bar{h}_{w,t}>\bar{h}_{v,t}+T} \min[1,\ \bar{h}_{w,t}-(\bar{h}_{v,t}+\delta_{v,t})-T] - \sum_{w:\,\bar{h}_{w,t}<\bar{h}_{v,t+1}-T} \min[1,\ (\bar{h}_{v,t}+\delta_{v,t})-\bar{h}_{w,t}-T]\right)\\
&\geq \bar{h}_{v,t+1} + \tfrac{1}{d}\left(\sum_{w:\,\bar{h}_{w,t}>\bar{h}_{v,t}+T} (\min[1,\ \bar{h}_{w,t}-\bar{h}_{v,t}-T]-\delta_{v,t}) - \sum_{w:\,\bar{h}_{w,t}<\bar{h}_{v,t+1}-T} (\min[1,\ \bar{h}_{v,t}-\bar{h}_{w,t}-T]+\delta_{v,t})\right)\\
&\geq \bar{h}_{v,t+1} - \delta_{v,t} + \tfrac{1}{d}\left(\sum_{w:\,\bar{h}_{w,t}>\bar{h}_{v,t}+T} \min[1,\ \bar{h}_{w,t}-\bar{h}_{v,t}-T] - \sum_{w:\,\bar{h}_{v,t}-T\leq\bar{h}_{w,t}<\bar{h}_{v,t+1}-T} \min[1,\ \bar{h}_{v,t}-\bar{h}_{w,t}-T]\right.\\
&\qquad \left. - \sum_{w:\,\bar{h}_{w,t}<\bar{h}_{v,t}-T} \min[1,\ \bar{h}_{v,t}-\bar{h}_{w,t}-T]\right)\\
&\geq \bar{h}_{v,t+1} - \delta_{v,t} + \tfrac{1}{d}\left(\sum_{w:\,\bar{h}_{w,t}>\bar{h}_{v,t}+T} \min[1,\ \bar{h}_{w,t}-\bar{h}_{v,t}-T] - \sum_{w:\,\bar{h}_{w,t}<\bar{h}_{v,t}-T} \min[1,\ \bar{h}_{v,t}-\bar{h}_{w,t}-T]\right)\\
&= \bar{h}_{v,t+1} - \delta_{v,t} + \delta_{v,t} = \bar{h}_{v,t+1}\ .
\end{aligned}
$$

Hence, also $\delta_{v,t+1} \geq 0$, which proves the lemma. □

This completes the proof of Theorem 3.11. □

In the fixpoint, the heights of the nodes do not change any more. This implies that, when the fixpoint is reached, the amount of flow entering a node is equal to the amount of flow leaving a node, i.e. the flow conservation principle is fulfilled. Since flow can only move from higher nodes to lower nodes, this implies that the flow movements form fixed, connected, and loop-free paths. The sum of the flow values of these paths is equal to the amount of flow injected into the system in every time step.

Recall that we assumed that $c(e) = 1$ and $w(e) = 1$ for every edge. In this case, the cost L_{BAL} of the flow paths found by the balancing algorithm is equal to

$$L_{BAL} = \sum_{\text{flow paths } p} \ell_p \cdot \lambda_p$$

where ℓ_p is the length of flow path p and λ_p is the amount of flow following path p. The question is how close L_{BAL} can be to a minimum-cost flow solution, L_{OPT}. The next theorem gives the answer to this question.

Theorem 3.13 *For any $T \geq 1$, the T-balancing algorithm ensures that, in the fixpoint, $L_{BAL} \leq (1 + 1/T)L_{OPT}$.*

Proof. We know that at the fixpoint of the balancing algorithm we have a fixed collection P of paths whose flow values sum up to the total injection rate. Let Q be any collection of optimal paths for the given injection process. W.l.o.g. we can assume that P and Q have the same number of paths, and the

ith path of P and Q has the same flow value and connects the same source-destination pair (otherwise split the paths of P and Q into subpaths such that this property is fulfilled). Thus, let $p_1, \ldots, p_r$ be the paths in P, and let $q_1, \ldots, q_r$ be the paths in Q. Let ℓ_i denote the length of p_i and k_i denote the length of q_i. Furthermore, let λ_i be the flow value of path p_i and q_i. Suppose now that

$$\sum_i \lambda_i \ell_i > \frac{T+1}{T} \sum_i \lambda_i k_i \ . \tag{1}$$

We will show via contradiction that this is not possible, which would prove the theorem.

For each $i \in \{1, \ldots, r\}$, let s_i denote the source of path p_i and q_i. Furthermore, for any node v let h_v denote the height of node v in the fixpoint, and for any edge $e = (v, w)$ let $\delta_e = h_v - h_w$. We define the *potential* of a collection C of paths p with flow values λ_p as

$$\Phi(C) = \sum_{p \in C} \lambda_p \sum_{(v,w) \in p} (h_v - h_w) \ .$$

Then we obtain for any collection of paths C connecting the same set of endpoints as P and Q that

$$\Phi(C) = \sum_i \lambda_i h_{s_i} \ .$$

Thus, $\Phi(P) = \Phi(Q)$. For any edge e and path collection C, let $\lambda_e(C)$ denote the amount of capacity of e used by the paths in C. Furthermore, for any two path collections C_1 and C_2 let $\lambda_e(C_1 \cap C_2) = \min[\lambda_e(C_1), \lambda_e(C_2)]$ and $\lambda_e(C_1 \setminus C_2) = \max[\lambda_e(C_1) - \lambda_e(C_2), 0]$. Then we obtain from $\Phi(P) = \Phi(Q)$ that

$$\sum_{e \in E} \delta_e \lambda_e(P \cap Q) + \sum_{e \in E} \delta_e \lambda_e(P \setminus Q) = \sum_{e \in E} \delta_e \lambda_e(P \cap Q) + \sum_{e \in E} \delta_e \lambda_e(Q \setminus P)$$

and thus

$$\sum_{e \in E} \delta_e \lambda_e(P \setminus Q) = \sum_{e \in E} \delta_e \lambda_e(Q \setminus P) \ .$$

We know that for all edges e with $\lambda_e(P \setminus Q) > 0$ we have $\delta_e \geq T \cdot d$, and for all edges e with $\lambda_e(Q \setminus P) > 0$ we have $\delta_e \leq (T+1)d$. Hence,

$$T \cdot d \sum_{e \in E} \lambda_e(P \setminus Q) \leq \sum_{e \in E} \delta_e \lambda_e(P \setminus Q)$$

and

$$\sum_{e \in E} \delta_e \lambda_e(Q \setminus P) \leq (T+1)d \sum_{e \in E} \lambda_e(Q \setminus P)$$

and so

$$\sum_{e \in E} \lambda_e(P \setminus Q) \leq \frac{T+1}{T} \sum_{e \in E} \lambda_e(Q \setminus P) \ .$$

On the other hand, (1) implies that

$$\sum_{e \in E} \lambda_e(P \cap Q) + \sum_{e \in E} \lambda_e(P \setminus Q) > \frac{T+1}{T} \left(\sum_{e \in E} \lambda_e(P \cap Q) + \sum_{e \in E} \lambda_e(Q \setminus P) \right)$$

and therefore

$$\sum_{e \in E} \lambda_e(P \setminus Q) > \frac{T+1}{T} \sum_{e \in E} \lambda_e(Q \setminus P) \ ,$$

which is a contradiction. □

Hence, the higher the T, the closer is the balancing algorithm to a minimum-cost flow solution. We believe that Theorem 3.13 can also be shown if the edges have non-uniform capacities and weights. However, in this case we have to change the definition of d to

$$d = \max_{v \in V} c(v)$$

where $c(v) = \sum_{w \in V} c(v, w)$ and the amount of flow forwarded along an edge $e = (v, w)$ to

$$c(e) \cdot \min\{(\bar{h}(v) - \bar{h}(w) - T \cdot w(e))/w(e), 1\} \ .$$

3.4 Multiple sources

Both the preflow algorithm and the T-balancing algorithm can be easily adapted to the situation that we have multiple sources (but still a single destination), say $s_1, \ldots, s_k$.

For the preflow algorithm, we just introduce a virtual source s in front of the sources $s_1, \ldots, s_k$ with edges (s, s_i) of capacity equalling the total capacity that can leave s_i. We then execute the algorithm for this situation, i.e. $s_1, \ldots, s_k$ are considered normal nodes.

For the balancing algorithm, we do not need any modifications for our results above to hold because at no place in our proofs we needed the fact that we only have a single source.

References

[1] B. Awerbuch, P. Berenbrink, A. Brinkmann, and C. Scheideler. Simple routing strategies for adversarial systems. In *Proc. of the 42nd IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 158–167, 2001.

[2] T. Corman, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press and McGraw-Hill, 2001.