Lecture 19: Max-Flow II

Michael Dinitz

November 2, 2021
601.433/633 Introduction to Algorithms

Introduction

Last time:

- Max-Flow = Min-Cut
- Can compute max flow and min cut using Ford-Fulkerson: while residual graph has an $\mathbf{s} \to \mathbf{t}$ path, push flow along it.
  - Corollary: if all capacities integers, max-flow is integral
  - If max-flow has value $\mathbf{F}$, time $\mathbf{O(F(m+n))}$ (if all capacities integers)
  - Exponential time!

Today:

- Important setting where FF is enough: max bipartite matching
- Two ways of making FF faster: Edmonds-Karp

Max Bipartite Matching

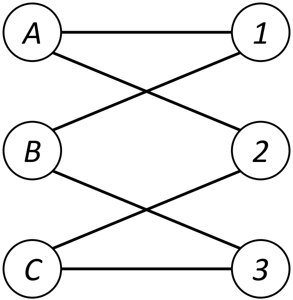
Setup

Definition

A graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is *bipartite* if $\mathbf{V}$ can be partitioned into two parts $\mathbf{L}, \mathbf{R}$ such that every edge in $\mathbf{E}$ has one endpoint in $\mathbf{L}$ and one endpoint in $\mathbf{R}$.

Definition

A *matching* is a subset $\mathbf{M} \subseteq \mathbf{E}$ such that $\mathbf{e} \cap \mathbf{e}' = \varnothing$ for all $\mathbf{e}, \mathbf{e}' \in \mathbf{M}$ with $\mathbf{e} \neq \mathbf{e}'$ (no two edges share an endpoint)



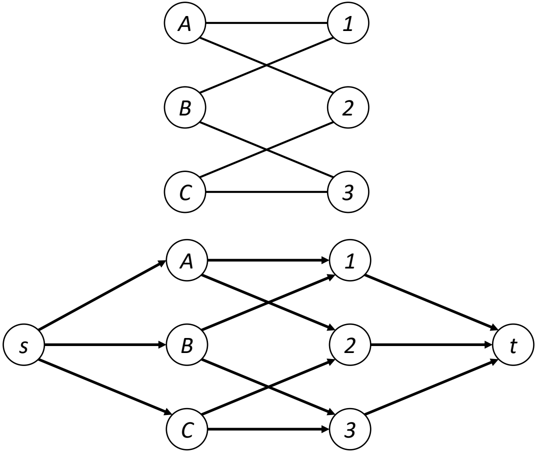
Bipartite Maximum Matching: Given bipartite graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, find matching $\mathbf{M}$ maximizing $|\mathbf{M}|$

- Extremely important problem, doesn't seem to have much to do with flow!

Algorithm

Give all edges capacity **1**
Direct all edges from **L** to **R**
Add source **s** and sink **t**
Add edges of capacity **1** from **s** to **L**
Add edges of capacity **1** from **R** to **t**

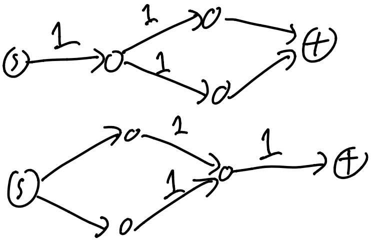
Run FF to get flow **f**
Return $\mathbf{M} = \{\mathbf{e} \in \mathbf{L} \times \mathbf{R} : \mathbf{f}(\mathbf{e}) > \mathbf{0}\}$



Correctness

Claim: $\mathbf{M}$ is a matching

Proof: capacities in $\{0, 1\} \implies \mathbf{f(e)} \in \{0, 1\}$ for all $\mathbf{e}$ (integrality)



Claim: $\mathbf{M}$ is maximum matching

Proof: Suppose larger matching $\mathbf{M}'$
Can send $|\mathbf{M}'|$ flow using $\mathbf{M}'$!

- $\mathbf{f'(s, u) = 1}$ is $\mathbf{u}$ matched in $\mathbf{M}'$, otherwise $\mathbf{0}$
- $\mathbf{f'(v, t) = 1}$ if $\mathbf{v}$ matched in $\mathbf{M}'$, otherwise $\mathbf{0}$
- $\mathbf{f'(u, v) = 1}$ if $\{\mathbf{u}, \mathbf{v}\} \in \mathbf{M}'$, otherwise $\mathbf{0}$
- $|\mathbf{f}'| = |\mathbf{M}'| > |\mathbf{M}| = |\mathbf{f}|$
- Contradiction

Running Time

Running Time:

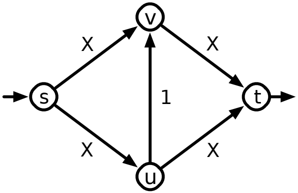
- $\mathbf{O(n+m)}$ to make new graph
- $|\mathbf{f}| = |\mathbf{M}| \leq \mathbf{n/2}$ iterations of FF

$\Longrightarrow$ $\mathbf{O(n(m+n)) = O(mn)}$ time (assuming $\mathbf{m \geq \Omega(n)}$)

Edmonds-Karp

Intuition

Bad example for Ford-Fulkerson:



A bad example for the Ford-Fulkerson algorithm.

If Ford-Fulkerson chooses bad augmenting paths, super slow!

Obvious idea: Choose better paths!

Obvious path to pick:

$$\operatorname*{arg\,max}_{\text{augmenting paths } P} \min_{e \in P} c_f(e).$$

- "Widest" path: push as much flow as possible each iteration

Edmonds-Karp #1

Edmonds-Karp #1: Ford-Fulkerson, always choose "widest" path.

- Correct, since FF. Running time?

Lemma

In any graph with max $\mathbf{s-t}$ *flow* $\mathbf{F}$*, there exists a path from* $\mathbf{s}$ *to* $\mathbf{t}$ *with capacity at least* $\mathbf{F/m}$

Proof.

Let $\mathbf{X} = \{\mathbf{e} \in \mathbf{E} : \mathbf{c(e)} < \mathbf{F/m}\}$.

If no $\mathbf{s \to t}$ path in $\mathbf{G \setminus X}$, then $\mathbf{X}$ an (edge) cut. Let $\mathbf{S}$ = nodes reachable from $\mathbf{s}$ in $\mathbf{G \setminus X}$.

$$\mathbf{cap(S, \bar{S}) \le cap(X) = \sum_{e \in X} c(e) < m \cdot (F/m) = F}$$

$\implies$ min $\mathbf{(s,t)}$ cut $\le \mathbf{cap(S,\bar{S})} < \mathbf{F}$. Contradiction.

$\implies$ $\exists \mathbf{s \to t}$ path $\mathbf{P}$ in $\mathbf{G \setminus X}$: every edge of $\mathbf{P}$ has capacity at least $\mathbf{F/m}$ $\square$

Does this implies at most $\mathbf{m}$ iterations?

EK 1 Running Time

Theorem

If $\mathbf{F}$ *is the value of the maximum flow and all capacities are integers, # iterations of EK1 is at most* $\mathbf{O(m \log F)}$

How much flow remains to be be sent after iteration $\mathbf{i}$?

- $\mathbf{i = 0}$: $\mathbf{F}$
- $\mathbf{I = 1}$: Sent at least $\mathbf{F/m}$, so at most $\mathbf{F - F/m = F(1 - 1/m)}$ remaining
- $\mathbf{i = 2}$: Sent at least $\mathbf{R/m}$ if $\mathbf{R}$ was remaining after iteration $\mathbf{1}$, so at most $\mathbf{R - R/m = R(1 - 1/m) \le F(1 - 1/m)^2}$ remaining

By induction: after iteration $\mathbf{i}$, at most $\mathbf{F(1 - 1/m)^i}$ flow remaining to be sent.
Super useful inequality: $\mathbf{1 + x \le e^x}$ for all $\mathbf{x} \in \mathbb{R}$
$\implies$ If $\mathbf{i > m \ln F}$, amount remaining to be sent at most

$$\mathbf{F(1 - 1/m)^i < F(1 - 1/m)^{m \ln F} \le F(e^{-1/m})^{m \ln F} = F \cdot e^{-\ln F} = 1}$$

But all capacities integers, so must be finished!

Finishing EK1

Modified version of Dijkstra: find widest path in $\mathbf{O(m \log n)}$ time

- Total time $\mathbf{O(m \log n \cdot m \log F) = O(m^2 \log n \log F)}$
- Polynomial time!

Question: can we get running time independent of $\mathbf{F}$?

- *Strongly* polynomial-time algorithm.

Edmonds-Karp #2

Again use Ford-Fulkerson, but pick *shortest* augmenting path (unweighted)

- Ignore capacities, just find augmenting path with fewest hops!
- Easy to compute with BFS in $\mathbf{O(m+n)}$ time.

Main question: how many iterations?

Theorem

EK2 has at most $\mathbf{O(mn)}$ *iterations, so at most* $\mathbf{O(m^2n)}$ *running time (if* $\mathbf{m \geq n}$*)*

Proof (sketch) of EK2

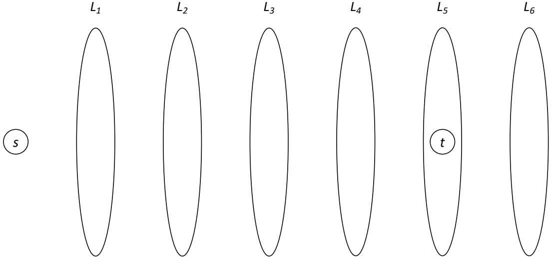
Idea: prove that distance from $\mathbf{s}$ to $\mathbf{t}$ (unweighted) goes up by at least one every $\leq \mathbf{m}$ iterations.

- Distance initially $\geq \mathbf{1} \implies$ distance $> \mathbf{n}$ after at most $\mathbf{mn}$ iterations
- Only distance larger than $\mathbf{n}$ is ∞: no $\mathbf{s} \to \mathbf{t}$ path

$\implies$ Terminates after at most $\mathbf{mn}$ iterations.

Proof (sketch) of EK2 (continued)

Suppose $\mathbf{s} \to \mathbf{t}$ distance is $\mathbf{d}$.
"Lay out" residual graph in levels by BFS (distance from $\mathbf{s}$)



Edge types:

- Forward edges: **1** level
- Edges inside level
- Backwards edges

What happens when we choose a *shortest* augmenting path? Only uses forward edges!

- At least **1** forward edge gets removed, replaced with backwards edge.
- No backwards edges turned forward

So after $\mathbf{m}$ iterations (same layout): no path using only forward edges $\Longrightarrow$ distance larger than $\mathbf{d}$!

Finishing EK2

So at most $\mathbf{mn}$ iterations. Each iteration unweighted shortest path: BFS, time $\mathbf{O(m+n)}$

Total time: $\mathbf{O(mn(m+n)) = O(m^2n)}$. Independent of $\mathbf{F}$!

Extensions

Many better algorithms for max-flow: *blocking flows* (Dinitz's algorithm (not me)), *push-relabel* algorithms, etc.

- CLRS has a few of these.
- State of the art:
  - Strongly polynomial: $\mathbf{O(mn)}$. Orlin [2013] & King, Rao, Tarjan [1994]
  - Weakly Polynomial: $\mathbf{\tilde{O}(m^{\frac{3}{2}-\frac{1}{328}} \log U)}$ (where $\mathbf{U}$ is maximum capacity). Gao, Liu, Peng [2021]

Many other variants of flows, some of which are just $\mathbf{s-t}$ max flow in disguise!

- Min-Cost Max-Flow: every edge also has a cost. Find minimum cost max-flow. Can be solved with just normal max flow!