Algorithms for Network Flows

Lecture 1: Maximum flows

Neil Olver

VU UNIVERSITY AMSTERDAM

CWI
Centrum Wiskunde & Informatica

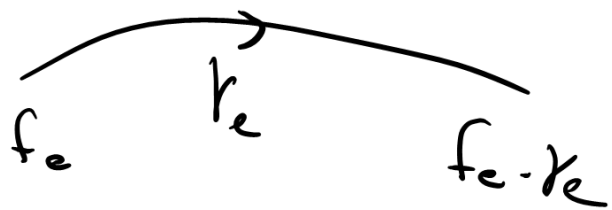
Valparaíso Summer School, 2017

Slides will be available at: `http://nolver.net/home/valparaiso`

What are these lectures about?

- Efficient, exact, “combinatorial” algorithms for flow problems
- Especially “strongly polynomial” algorithms

- **Lecture 1:** Maximum flow
- **Lecture 2:** Minimum cost flow
- **Lectures 3 and 4:** Generalized flows



Today's lecture

- Two basic polynomial-time max flow algorithms
- The notion of strongly polynomial time
- A faster (and strongly polynomial) algorithm

Maximum flow

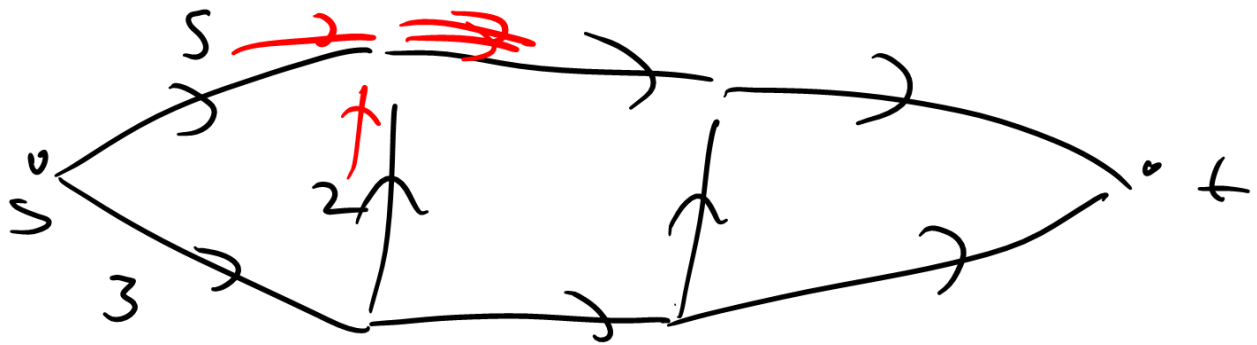
Given: Directed graph $G = (V, E)$, edge capacities $u : E \to \mathbb{R}_+$, source $s \in V$, sink $t \in V$
Goal: Find an s-t-flow of maximum value.

- An s-t-flow is a function $f : E \to \mathbb{R}_+$ s.t.

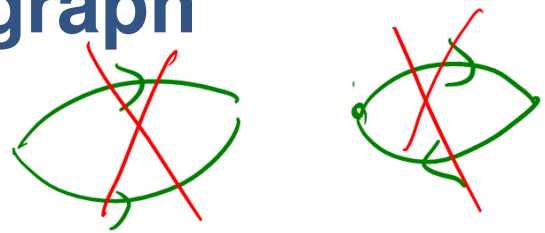
$$\nabla f_i := \sum_{e \in \delta^-(i)} f(e) - \sum_{e \in \delta^+(i)} f(e) = 0 \quad \forall i \in V \setminus \{s, t\}.$$

The value of the flow is

$$\text{value}(f) := \nabla f_t = -\nabla f_s.$$



Augmenting paths and the residual graph



- Assume E has no parallel edges or digons.

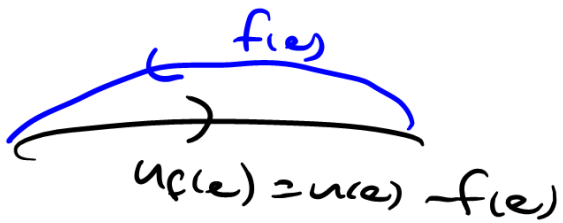
Let

$$\overleftrightarrow{E} := E \cup \{\mathrm{rev}(e) : e \in E\}.$$

For $f : E \to \mathbb{R}_+$ with $0 \le f \le u$, define $u_f : \overleftrightarrow{E} \to \mathbb{R}_+$ by

$$u_f(e) = \begin{cases} u(e) - f(e) & \text{if } e \in E \\ f(\mathrm{rev}(e)) & \text{if } \mathrm{rev}(e) \in E \end{cases}$$

Let $E_f = \mathrm{supp}(u_f) \subseteq \overleftrightarrow{E}$.



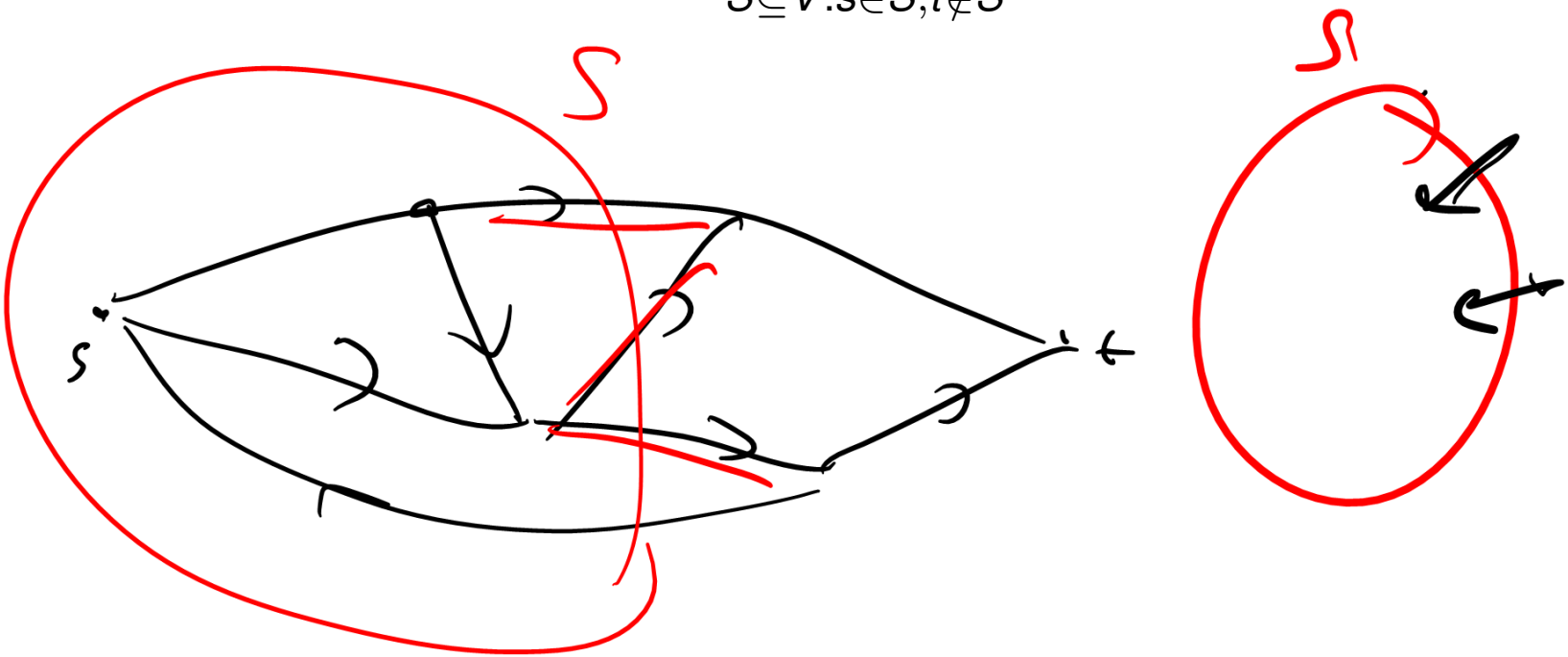
Some reminders

Max-flow min-cut

$$\max_{\text{flows } f} \text{value}(f) = \min_{S \subseteq V : s \in S, t \notin S} u(\delta^+(S)).$$

- Given any flow f,

$$\text{value}(f^*) = \text{value}(f) + \min_{S \subseteq V : s \in S, t \notin S} u_f(\delta^+(S)).$$



Some reminders

Max-flow min-cut

$$\max_{\text{flows } f} \text{value}(f) = \min_{S \subseteq V : s \in S, t \notin S} u(\delta^+(S)).$$

- Given any flow f,

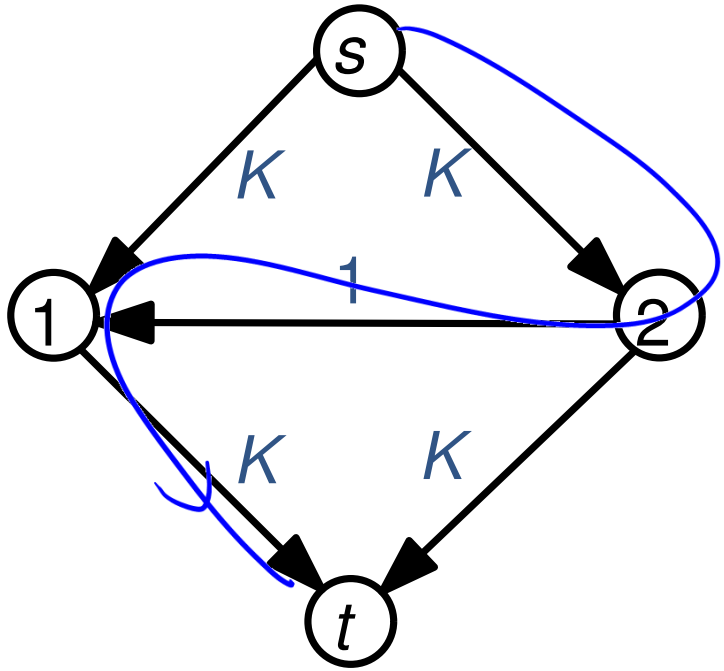
$$\text{value}(f^*) = \text{value}(f) + \min_{S \subseteq V : s \in S, t \notin S} u_f(\delta^+(S)).$$

Flow decomposition

If f is a flow, then $f = \sum_{i=1}^{k} \lambda_i \chi(P_i)$, where $k \leq m$, $\lambda_i \geq 0$, P_i is either an s-t-path or a cycle.

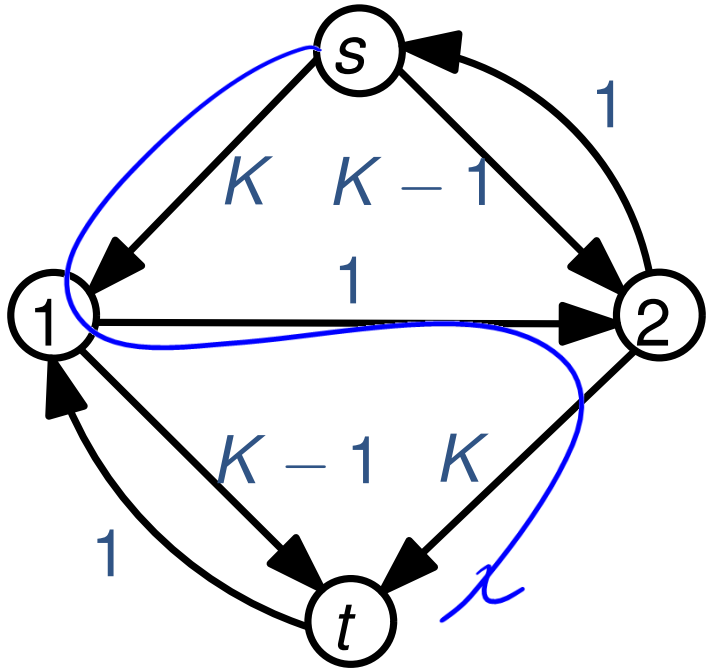
Ford-Fulkerson isn't polynomial

1: **while** $\exists$ s-t path P in E_f **do**
2: $\quad f \leftarrow \text{augment}(f, P)$



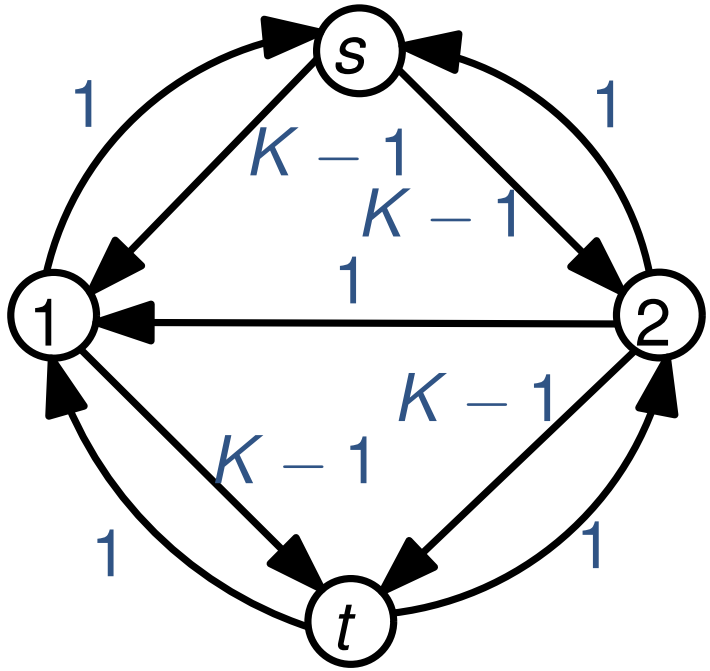
Ford-Fulkerson isn't polynomial

1: **while** $\exists$ s-t path P in E_f **do**
2: $\quad f \leftarrow \text{augment}(f, P)$



Ford-Fulkerson isn't polynomial

1: **while** $\exists$ s-t path P in E_f **do**
2: $\quad f \leftarrow \text{augment}(f, P)$



Capacity scaling

Edmonds-Karp '72 / Ahuja-Orlin '91

Idea: Try to use paths where we can send a lot of flow.

- For this algorithm, we'll assume **integer capacities**.
- Define, for any $\Delta > 0$, $E_f^\Delta := \{e \in E_f : u_f(e) \geq \Delta\}$.

1: $U := \max_{e \in E} u(e)$, $f \leftarrow 0$
2: $\Delta \leftarrow 2^{\lfloor \log_2 U \rfloor}$
3: **while** $\Delta \geq 1$ **do**
4: **while** $\exists$ s-t path P in E_f^Δ **do**
5: $f \leftarrow \text{augment}(f, P)$
6: $\Delta \leftarrow \Delta/2$

Analysis

Lemma

This algorithm returns a maximum flow.

Pf: $E_f^1 = E_f$ (since c integral, and we maintain that f is integral also)

$\therefore$ At termination, E_f has no s-t-paths,

so f maximum. ∎

Lemma

Each phase (where Δ is constant) has at most $2m$ augmentations.

Pf: At start of phase, $E_f^{2\Delta}$ contains no s-t paths

$\therefore \text{val}(f^m) \leq \text{val}(f) + m \cdot 20$

Each aug. increases $\text{val}(f)$ by $\Delta \cdot 10$.

$O(\log U)$

Each augmentation can be done in $O(m)$

$\rightsquigarrow O(m^2 \log U)$ time.

Second algorithm: shortest paths

Edmonds-Karp '72

1: **while** $\exists$ s-t path in E_f **do**
2: Choose P to be a shortest s-t-path in E_f
3: $f \leftarrow \text{augment}(f, P)$

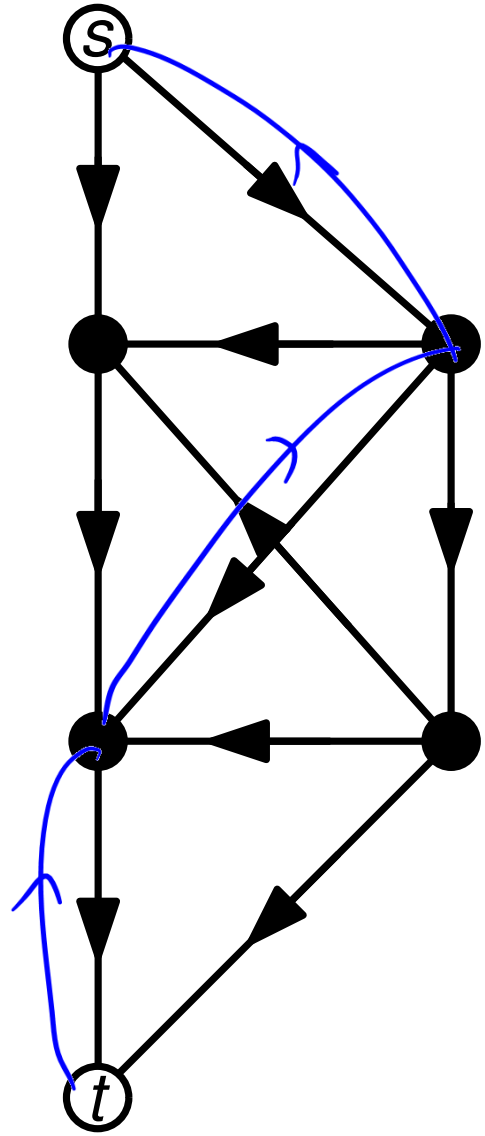
$f^{(r)}$ - flow after r iterations

$d_r(i)$ - shortest path dist from i to t in $E_{f^{(r)}}$.

Claim: $d_{r+1}(i) \geq d_r(i)$ $\forall r, i$

Phase: Sequence of iterations where $d_r(s)$ is unchanged.

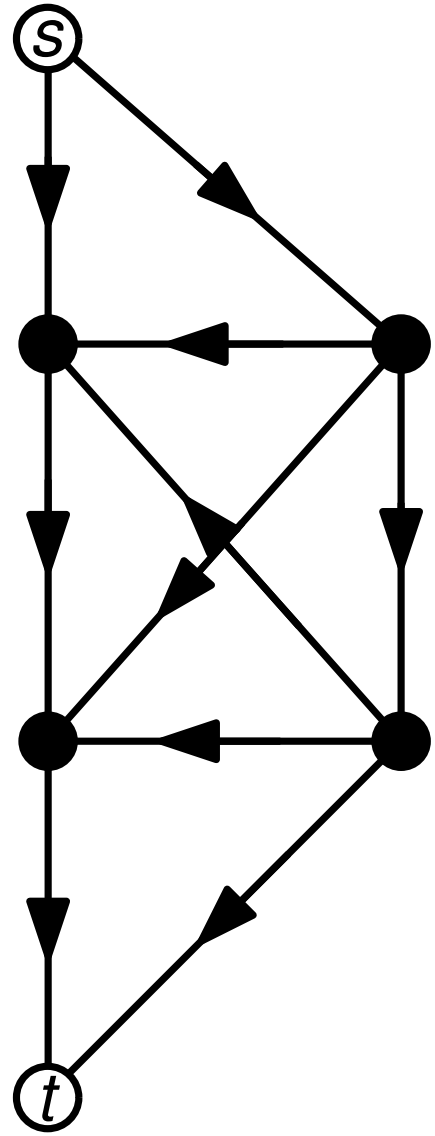
At most n phases.



Claim: $\leq m$ iterations in a phase.

n phases, m iterations per phase, $O(m)$ operations per iter.

$\rightsquigarrow$ $O(nm^2)$.



Strongly and weakly polynomial

- Weakly polynomial: polynomial time in the Turing model

An algorithm is strongly polynomial if:

1. Number of arithmetic operations is polynomial in the number of integers in the input (e.g., size of the graph)
2. Encoding lengths of numbers computed during execution are polynomial in input encoding length

Given k, compute 2^k. ← NOT strongly polynomial, because 2 fn. Ks. Not polytime at all!

Given a, b, compute $\gcd(a, b)$ ← No strongly poly. alg.

Given $k \in \mathbb{N}$, compute the largest power of 2 $\leq k$.

Why care about strongly polynomial?

- Always faster in some parameter regimes (huge capacities)
- Finding a strongly polynomial algorithm often requires new, useful insights
- Performance of weakly polynomial algorithms tied up with the particular “standard” representation of integers and rationals

A holy grail

"Strongly polynomial linear programming has been a holy grail for the theory of algorithms for several decades."

Bárász and Vempala

- One of Smale's 18 problems

A holy grail

"Strongly polynomial linear programming has been a holy grail for the theory of algorithms for several decades."

Bárász and Vempala

- One of Smale's 18 problems
- Consider an LP

$$\min c^T x \quad \text{s.t.} \quad Ax = b, x \geq 0,$$

where $A \in \{0, 1, -1\}^{mn}$ and each column of A has one +1 and one -1 entry.

- This corresponds to min cost flow.
- The question of feasibility corresponds to max flow.

A faster strongly polynomial algorithm

Definition

A preflow is a function $f : E \to \mathbb{R}_+$ where

$$f(e) \leq u(e) \qquad \forall e \in E$$
$$\nabla f_i \geq 0 \qquad \forall i \in V \setminus \{s, t\}$$

$m = |E|$

$n = |V|$

- A node $i \in V$ is active if $v \notin \{s, t\}$ and $\nabla f_i > 0$.

$d : V \to \mathbb{Z}$ is called a valid distance labelling for f if $d(t) = 0$, $d(i) \leq d(j) + 1$ for $ij \in E_f$

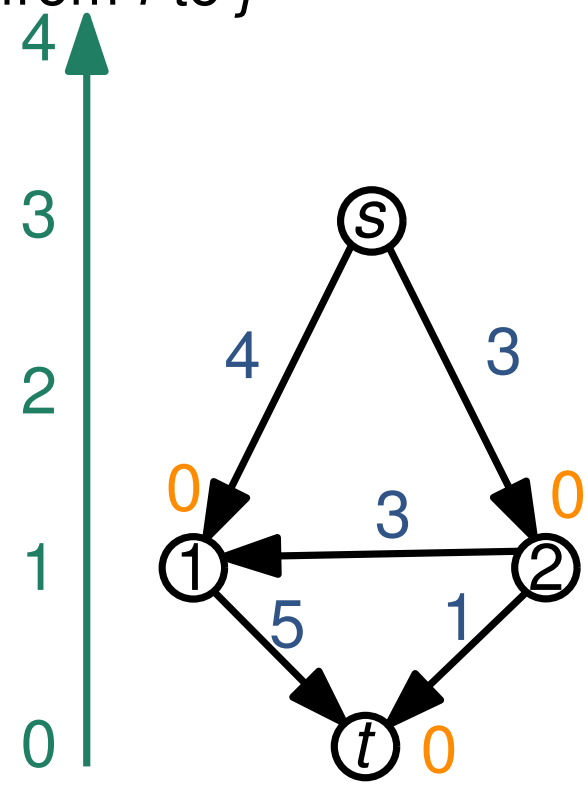
- An edge $ij \in \overleftrightarrow{E}$ is admissable (w.r.t. d) if $d(j) = d(i) - 1$ and $ij \in E_f$.

If d valid $\Rightarrow$ $d(i) \leq$ S.P. distance to t in E_f

Push-relabel

Goldberg-Tarjan '88

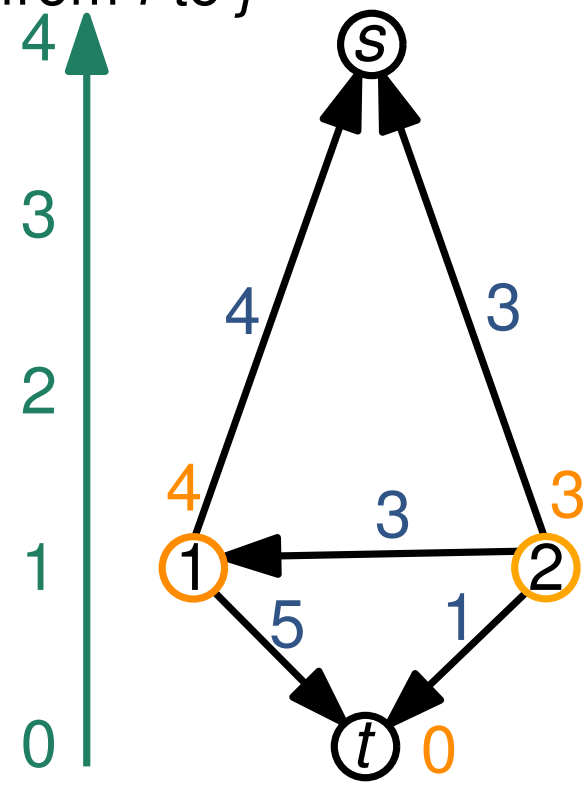
1: $d(i) \leftarrow \mathrm{SP}_E(i,t) \quad \forall i \in V$
2: $f(e) = u(e) \quad \forall e \in \delta^+(s);\ d(s) \leftarrow n$
3: **while** $\exists$ an active node **do**
4: Pick i to be an active node with largest $d(i)$
5: **if** $\exists j$ s.t. ij admissable **then**
6: Push $\delta := \min\{\nabla f_i, u_f(ij)\}$ units of flow from i to j
7: **else**
8: $d(i) \leftarrow \min\{d(k) + 1 : ik \in E_f\}$



Push-relabel

Goldberg-Tarjan '88

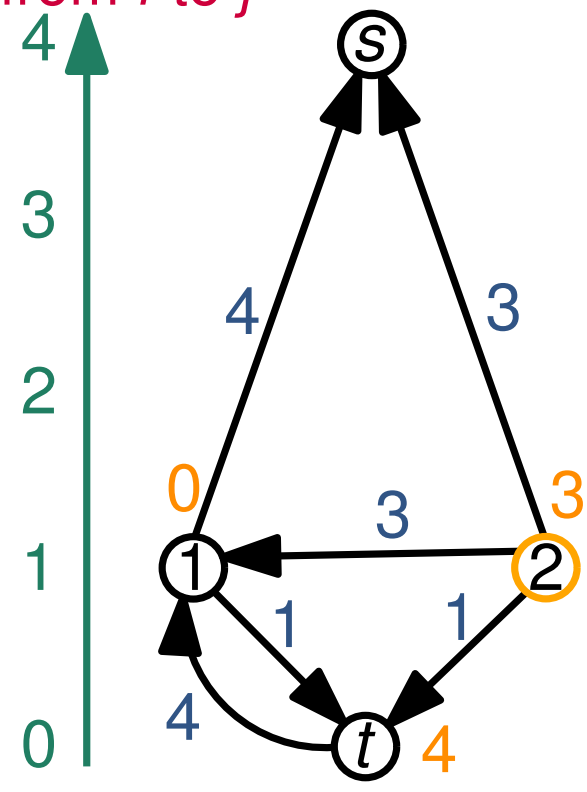
1: $d(i) \leftarrow \mathrm{SP}_E(i,t) \quad \forall i \in V$
2: $f(e) = u(e) \quad \forall e \in \delta^+(s);\ d(s) \leftarrow n$
3: **while** $\exists$ an active node **do**
4: Pick i to be an active node with largest $d(i)$
5: **if** $\exists j$ s.t. ij admissable **then**
6: Push $\delta := \min\{\nabla f_i, u_f(ij)\}$ units of flow from i to j
7: **else**
8: $d(i) \leftarrow \min\{d(k) + 1 : ik \in E_f\}$



Push-relabel

Goldberg-Tarjan '88

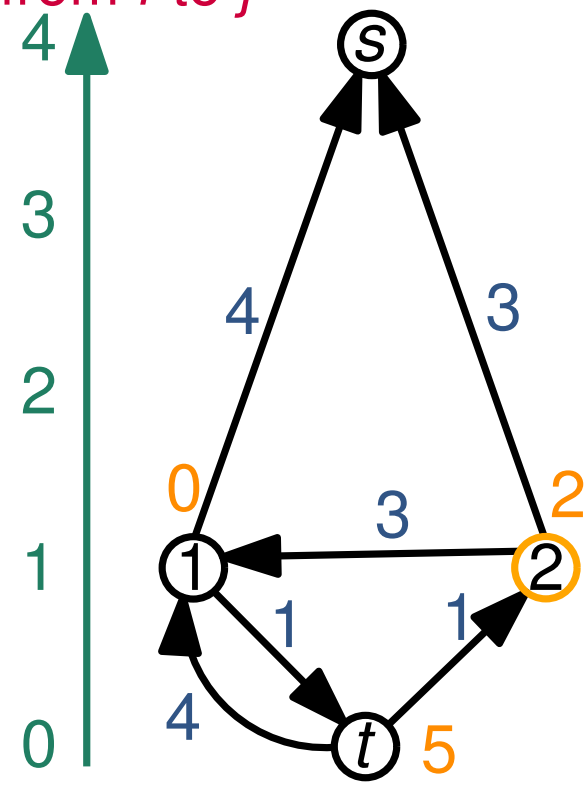
1: $d(i) \leftarrow \mathrm{SP}_E(i,t) \quad \forall i \in V$
2: $f(e) = u(e) \quad \forall e \in \delta^+(s);\ d(s) \leftarrow n$
3: **while** $\exists$ an active node **do**
4: Pick i to be an active node with largest $d(i)$
5: **if** $\exists j$ s.t. ij admissable **then**
6: Push $\delta := \min\{\nabla f_i, u_f(ij)\}$ units of flow from i to j
7: **else**
8: $d(i) \leftarrow \min\{d(k) + 1 : ik \in E_f\}$



Push-relabel

Goldberg-Tarjan '88

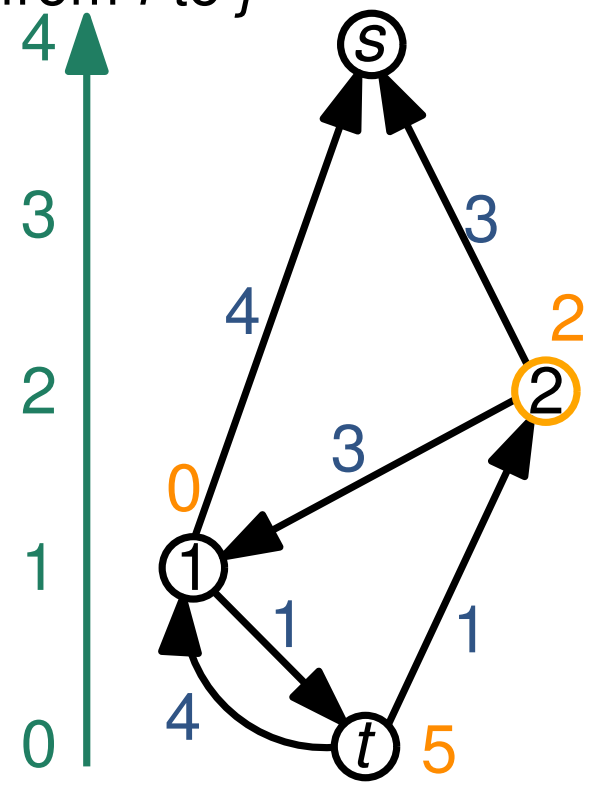
1: $d(i) \leftarrow \mathrm{SP}_E(i,t) \quad \forall i \in V$
2: $f(e) = u(e) \quad \forall e \in \delta^+(s);\ d(s) \leftarrow n$
3: **while** $\exists$ an active node **do**
4: Pick i to be an active node with largest $d(i)$
5: **if** $\exists j$ s.t. ij admissable **then**
6: Push $\delta := \min\{\nabla f_i, u_f(ij)\}$ units of flow from i to j
7: **else**
8: $d(i) \leftarrow \min\{d(k) + 1 : ik \in E_f\}$



Push-relabel

Goldberg-Tarjan '88

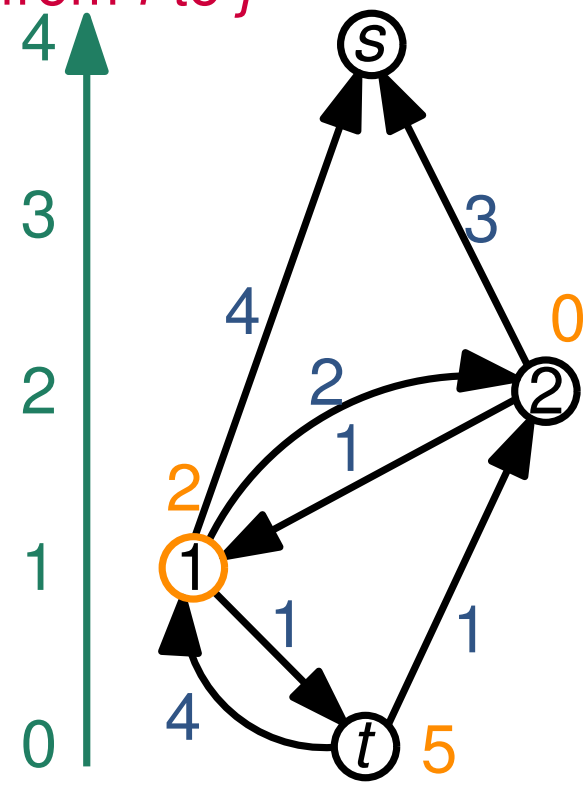
1: $d(i) \leftarrow \mathrm{SP}_E(i,t) \quad \forall i \in V$
2: $f(e) = u(e) \quad \forall e \in \delta^+(s);\ d(s) \leftarrow n$
3: **while** $\exists$ an active node **do**
4: Pick i to be an active node with largest $d(i)$
5: **if** $\exists j$ s.t. ij admissable **then**
6: Push $\delta := \min\{\nabla f_i, u_f(ij)\}$ units of flow from i to j
7: **else**
8: $d(i) \leftarrow \min\{d(k)+1 : ik \in E_f\}$



Push-relabel

Goldberg-Tarjan '88

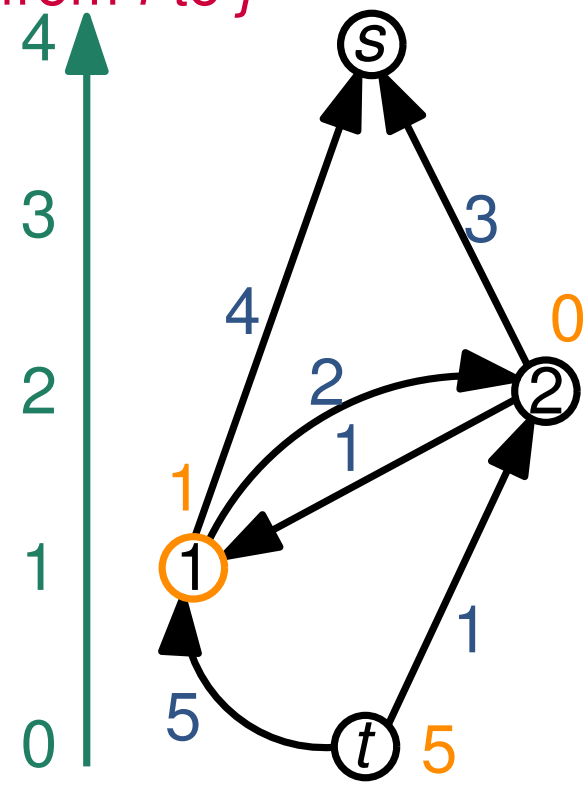
1: $d(i) \leftarrow \mathrm{SP}_E(i,t) \quad \forall i \in V$
2: $f(e) = u(e) \quad \forall e \in \delta^+(s);\ d(s) \leftarrow n$
3: **while** $\exists$ an active node **do**
4: Pick i to be an active node with largest $d(i)$
5: **if** $\exists j$ s.t. ij admissable **then**
6: Push $\delta := \min\{\nabla f_i, u_f(ij)\}$ units of flow from i to j
7: **else**
8: $d(i) \leftarrow \min\{d(k) + 1 : ik \in E_f\}$



Push-relabel

Goldberg-Tarjan '88

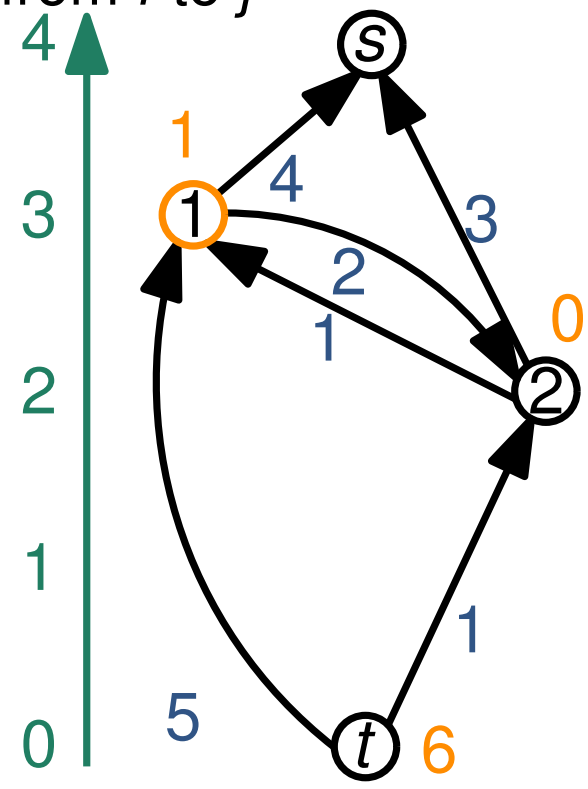
1: $d(i) \leftarrow \mathrm{SP}_E(i,t) \quad \forall i \in V$
2: $f(e) = u(e) \quad \forall e \in \delta^+(s);\ d(s) \leftarrow n$
3: **while** $\exists$ an active node **do**
4: Pick i to be an active node with largest $d(i)$
5: **if** $\exists j$ s.t. ij admissable **then**
6: Push $\delta := \min\{\nabla f_i, u_f(ij)\}$ units of flow from i to j
7: **else**
8: $d(i) \leftarrow \min\{d(k) + 1 : ik \in E_f\}$



Push-relabel

Goldberg-Tarjan '88

1: $d(i) \leftarrow \mathrm{SP}_E(i,t) \quad \forall i \in V$
2: $f(e) = u(e) \quad \forall e \in \delta^+(s);\ d(s) \leftarrow n$
3: **while** $\exists$ an active node **do**
4: Pick i to be an active node with largest $d(i)$
5: **if** $\exists j$ s.t. ij admissable **then**
6: Push $\delta := \min\{\nabla f_i, u_f(ij)\}$ units of flow from i to j
7: **else**
8: $d(i) \leftarrow \min\{d(k)+1 : ik \in E_f\}$



Lemma

At any point in the algorithm, d is a valid labelling.

Lemma

If the push-relabel algorithm terminates, it terminates with a max flow.

pf: No active nodes $\rightarrow$ $Df_i = 0$ $\forall i \neq s,t$
$\therefore$ flow.

d valid, $d(s) = n$, $d(t) = 0$.

$\Rightarrow$ no s-t-path in E_f, since any such path has length $< n$.

$\therefore$ f is maximum. ☐

Lemma

Throughout the algorithm, $\exists$ i-s-path in E_f for any active i.

Pf $\exists$ s-i-path P in $\mathrm{supp}(f)$.

$\mathrm{rev}(P) \subseteq E_f$.

Lemma

$d(i) < 2n$ for all $i \in V$. Hence the algorithm performs at most $2n^2$ relabellings.

Pf: Last time we relabel i, $\exists$ i-s-path Q in E_f.

$d(i) \le d(s) + |E(Q)|$
$< 2n$.

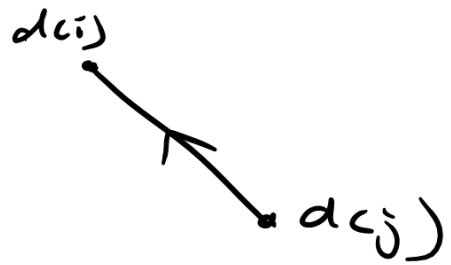
1: $d(i) \leftarrow \mathrm{SP}_E(i,t) \quad \forall i \in V$
2: $f(e) = u(e) \quad \forall e \in \delta^+(s); d(s) \leftarrow n$
3: **while** $\exists$ an active node **do**
4: Pick i to be an active node with largest $d(i)$
5: **if** $\exists j$ s.t. ij admissable **then**
6: Push $\delta := \min\{\nabla f_i, u_f(ij)\}$ units of flow from i to j
7: **else**
8: $d(i) \leftarrow \min\{d(k) + 1 : ik \in E_f\}$

Call a push operation saturating if $\delta = u_f(ij)$, nonsaturating otherwise.

Lemma

There are at most mn saturating pushes.

Pf: Say ij saturated at iteration r.



$d(j)$ must increase by 2 before we can unsaturate the arc

$d(j) < 2n$, so ij saturated at most n times

$\forall$

Lemma

There are at most $O(n^3)$ nonsaturating pushes.

pf: "Phase": sequence of iterations where no relabels performed.

($< 2n^2$ phases)

A nonsaturating push on ij deactivates i. Each i can be deactivated at most once per phase, since all remaining active nodes are lower. ∎

$O(n^3)$ iterations.

Can be implemented in $O(n^3)$.

$\rightsquigarrow$ $O(n^3)$ operations

State of the art for max flow

- **Fastest strongly polynomial algorithm:** $O(mn)$ Orlin '13 (using King-Rao-Tarjan '94 and Goldberg-Rao '98)

- **Fastest weakly polynomial algorithm:** $\tilde{O}(m\sqrt{n}\,\text{polylog}(U))$ Lee-Sidford '13

  Befre then, combinatorial algorithm of Goldberg-Rao '98

Unit capacities: $\tilde{O}(m^{10/7})$ Mądry '13

- **ϵ-approximate maximum flows in undirected graphs:** $\tilde{O}(m)$ Peng '16, building on many papers using electrical networks Christiano et al., Lee et al., Sherman, Kelner et al., ...

Exercise

Consider a variant of the push-relabel algorithm where instead of choosing the active node with largest label, we choose the active node with smallest label.

Prove that the number of nonsaturating pushes is bounded by $O(mn^2)$. (This is a worse bound than what we achieved with the largest label, but still strongly polynomial).

Hint: use a potential function that involves the labels of the active nodes.

References


R. K. Ahuja, T. L. Magnanti, and J. B. Orlin.
Network Flows: Theory, Algorithms, and Applications.
Prentice-Hall, 1993.


A. V. Goldberg and R. E. Tarjan.
Efficient maximum flow algorithms.
Communications of the ACM, 2015.


J. B. Orlin.
Max flows in $O(nm)$ time or better.
In *Proceedings of STOC*, 2013.