Advanced Algorithms

Nicole Megow (Universität Bremen)

SoSe 2025

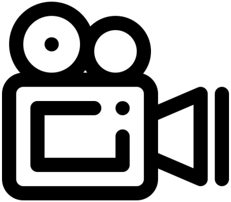
Network Flows: Blocking Flows

Lecture 6

Recording of this Lecture

This lecture will be recorded

- Recording only of the lecturers by themselves.
- If there are questions from the audience, please make a clear signal if the microphone shall be muted.
- Our goal is to record the lecture, but it is no guarantee that each lecture will be recorded.

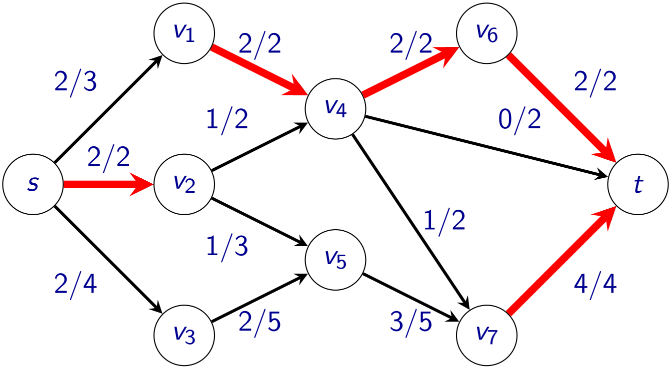


Last Lecture

- Introduction to network flows
- Optimality criterion of augmenting paths
- Max-Flow Min-Cut Theorem
- Ford-Fulkerson algorithm: $O(m \cdot M)$
- Edmonds-Karp algorithm: $O(nm^2)$

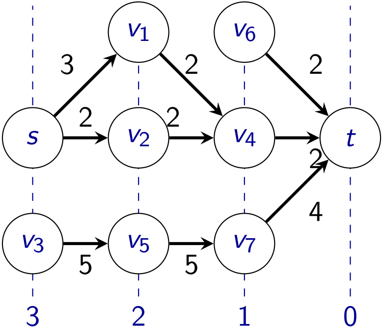
Blocking Flows

A blocking flow in an s-t-network $\mathcal{N} = (V, E, c, s, t)$ is a feasible flow that saturates at least one edge on every s-t-path in $\mathcal{N}$.



Layered Networks

Let $\mathcal{N} = (V, E, c, s, t)$ be an s-t-network. The layered network of $\mathcal{N}$ w.r.t. f is the s-t-network $\mathcal{N}' = (V, E', c', s, t)$, where $E' = \{(u, v) \in E_f \mid \delta_f(v) < \delta_f(u)\}$ and c' is the restriction of c_f to E'. For an $i \in \mathbb{N}_0$, the layer i of $\mathcal{N}'$ is the set of vertices $V_i = \{v \in V \mid \delta_f(v) = i\}$.



Computing Blocking Flows

Lemma

A blocking flow in an acyclic s-t-network $\mathcal{N} = (V, E, c, s, t)$ with n vertices and m edges be computed in $O(nm)$ time.

We use the following algorithm:

- Repeatedly start a DFS from s until reaching t.
- Whenever the DFS retreats over an edge e, remove e.
- Whenever reaching t, carry out an augmentation along the found s-t-path. Remove all edges that become saturated by the augmentation.
- Stop when t is unreachable from s.

Dinitz' Algorithm

1. $f \leftarrow 0$
2. As long as f is not maximum, compute a blocking flow h in the layered network of $\mathcal{N}_f$ w.r.t. f, and set $f \leftarrow f + h$
3. Return f

Lemma

Every iteration of Dinitz' algorithm strictly increases $\delta_f(s)$.

Theorem

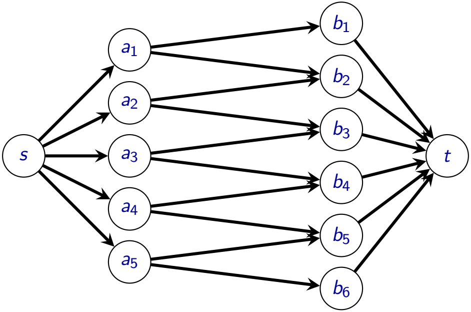
Dinitz' algorithm computes a maximum flow in an s-t-network with n vertices and m edges in $O(n^2 m)$ time.

Edmond-Karp's algorithm only runs in $O(nm^2)$ time.

Simple s-t-Networks

An s-t-network $\mathcal{N} = (V, E, c, s, t)$ is simple if all capacities are integers, and for every $v \in V \setminus \{s, t\}$, at least one of the following conditions holds:

1. the capacities of the edges entering v sum up to at most 1, or
2. the capacities of the edges leaving v sum up to at most 1.



Universität Bremen

Dinitz' Algorithm for Simple Networks

Lemma

Dinitz' algorithm terminates after at most $O(\sqrt{n})$ iterations when applied to a simple s-t-network.

Lemma

A blocking flow in the layered network $\mathcal{L}$ of a simple s-t-network $\mathcal{N}$ w.r.t. an integral flow f in $\mathcal{N}$ can be computed in $O(n+m)$ time.

Theorem

Dinitz' algorithm computes a maximum flow in a simple s-t-network with n vertices and m edges in $O(m \cdot \sqrt{n})$ time.

A maximum matching in a bipartite graph with n vertices and m edges can be computed in $O(m \cdot \sqrt{n})$ time.