# Flows, matchings and applications

Jill-Jênn Vie (Erickson, 2019)

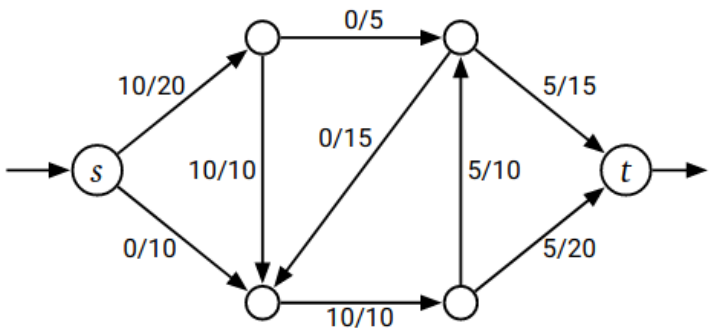
Oct 28, 2022

# Flows

Let $G = (V, E)$ a graph with capacities $c : E \to \mathbf{R}^+$ on edges.

A flow $f : E \to \mathbf{R}$ should verify for all $v \in V \setminus \{\underbrace{s}_{\text{source}}, \underbrace{t}_{\text{sink}}\}$ the conservation of flow:

$$\underbrace{\sum_u f(u \to v)}_{\text{flow arriving in } v} = \underbrace{\sum_w f(v \to w)}_{\text{flow out of } v}.$$

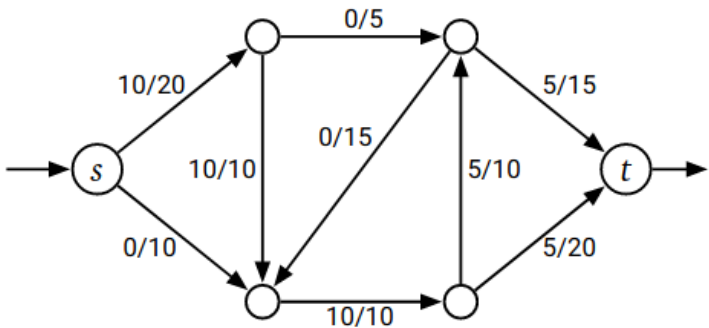


# Max-flow

Let $G = (V, E)$ a graph with capacities $c : E \to \mathbf{R}^+$ on edges.

A flow $f : E \to \mathbf{R}$ should verify for all $v \in V \setminus \{s, t\}$: $\sum_u f(u \to v) = \sum_w f(v \to w)$ and for all $e \in E$, $f(e) \leq c(e)$.
The value of flow $|f|$ is the total flow out of source $s$ = received by $t$:

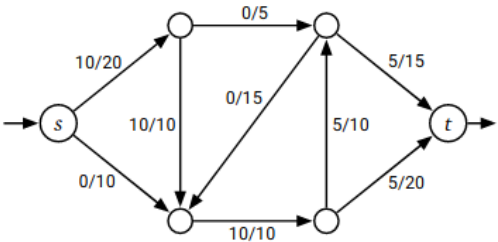
$$|f| = \sum_w f(s \to w) = \sum_u f(u \to t)$$

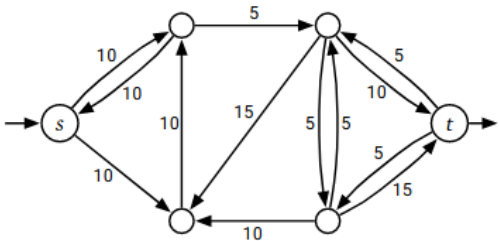


# Residual graph

We define the residual capacity

$$c_f : u \to v \mapsto \begin{cases} c(u \to v) - f(u \to v) & \text{if } u \to v \in E \\ f(v \to u) & \text{if } v \to u \in E \\ 0 & \text{otherwise} \end{cases}$$

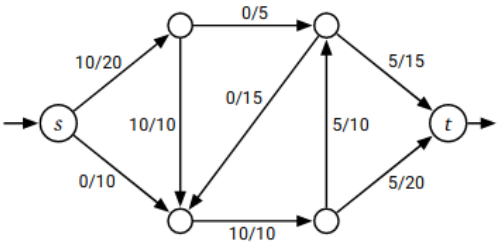


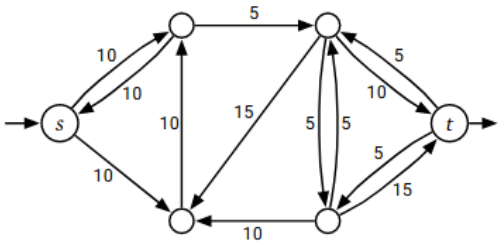



# Residual graph

We define the residual capacity

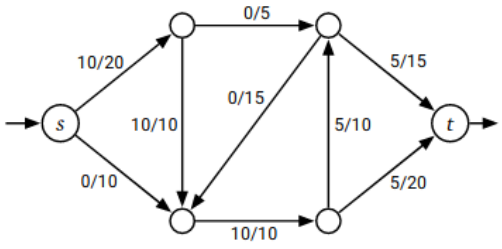
$$c_f : u \to v \mapsto \begin{cases} c(u \to v) - f(u \to v) & \text{if } u \to v \in E \\ f(v \to u) & \text{if } v \to u \in E \\ 0 & \text{otherwise} \end{cases}$$

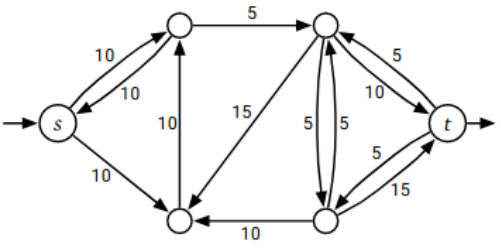


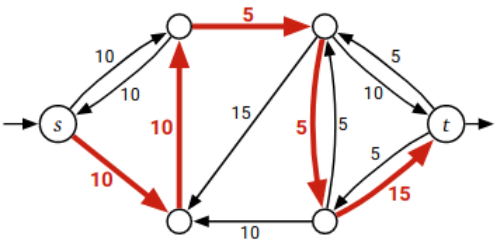



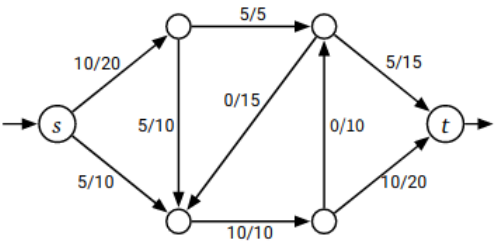

$|f|$ is not maximum $\iff$ there exists an augmenting path

# Max-flow


0/5
10/20
5/15
0/15
s
10/10
5/10
t
0/10
5/20
10/10


5
10
5
10
10
15
10
s
5
5
t
5
10
15
10


5
10
5
10
10
15
10
s
5
5
t
10
5
5
10
15


5/5
10/20
5/15
0/15
s
5/10
0/10
t
5/10
10/20
10/10

# Max-flow algorithms

The way the augmenting path is found has an incidence on the algorithm complexity.

- Any path (e.g. DFS): Ford-Fulkerson algorithm $O(E|f^*|)$
- Path with largest bottleneck value (e.g. Dijkstra): $O(E^2 \log E \log |f^*|)$
- Path with smallest number of edges (BFS): Edmonds-Karp $O(VE^2)$
- Dinitz: $O(V^2E)$

# Max-flow algorithms

The way the augmenting path is found has an incidence on the algorithm complexity.

- Any path (e.g. DFS): Ford-Fulkerson algorithm $O(E|f^*|)$
- Path with largest bottleneck value (e.g. Dijkstra): $O(E^2 \log E \log |f^*|)$
- Path with smallest number of edges (BFS): Edmonds-Karp $O(VE^2)$
- Dinitz: $O(V^2E)$

## Min-cost max-flow

We also have costs on edges $w : E \to \mathbf{R}$.
For each edge $u \to v$ we add $w(v \to u) = -w(u \to v)$ and $c(v \to u) = 0$.

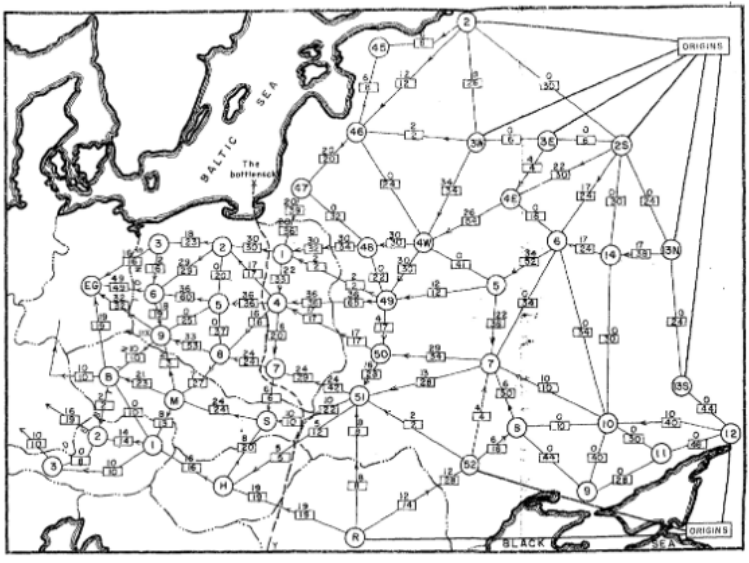
- Path with lowest cost for $w$ with Bellman-Ford: $O(V^2E^2)$

# And many more

## History of Worst-Case Running Times

| Year | Discoverer | Method | Asymptotic Time |
|---|---|---|---|
| 1951 | Dantzig | Simplex | $E\,V^2\,U^\dagger$ |
| 1955 | Ford, Fulkerson | Augmenting path | $E\,V\,U^\dagger$ |
| 1970 | Edmonds-Karp | Shortest path | $E^2\,V$ |
| 1970 | Edmonds-Karp | Max capacity | $E \log U\,(E + V \log V)^\dagger$ |
| 1970 | Dinitz | Improved shortest path | $E\,V^2$ |
| 1972 | Edmonds-Karp, Dinitz | Capacity scaling | $E^2 \log U^\dagger$ |
| 1973 | Dinitz-Gabow | Improved capacity scaling | $E\,V \log U^\dagger$ |
| 1974 | Karzanov | Preflow-push | $V^3$ |
| 1983 | Sleator-Tarjan | Dynamic trees | $E\,V \log V$ |
| 1986 | Goldberg-Tarjan | FIFO preflow-push | $E\,V \log (V^2 / E)$ |
| ... | ... | ... | ... |
| 1997 | Goldberg-Rao | Length function | $E^{3/2} \log (V^2 / E) \log U^\dagger$<br>$EV^{2/3} \log (V^2 / E) \log U^\dagger$ |

# Historical motivation




RM-3373
10-24-55
SECRET -53-

Fig. 7 — Traffic pattern: entire network available

Legend:

International boundary

Railway operating division

Capacity: 12 each way per day. Required flow of 9 per day toward destinations (in direction of arrow) with equivalent number of returning trains in opposite direction

All capacities in {trains / 1000's of tons} each way per day

Origins: Divisions 2, 3W, 3E, 25, 13N, 13S, 12, 52 (USSR), and Roumania

Destinations: Divisions 3, 6, 9 (Poland); 8 (Czechoslovakia); and 2, 3 (Austria)

Alternative destinations: Germany or East Germany

Note IIK of Division 9, Poland

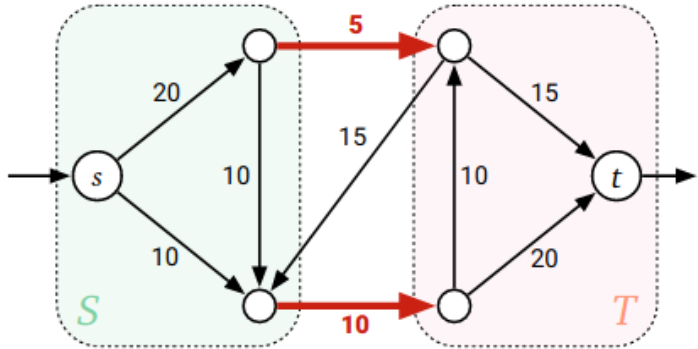
# Application: Min-cut

An cut is a partition of vertices into $S \ni s$ and $T \ni t$.

The capacity of a cut is

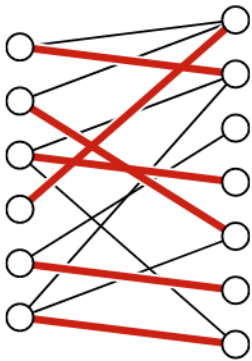
$$||S, T|| = \sum_{v \in S} \sum_{w \in T} c(v \to w).$$

How to find a cut of minimal capacity?



# Application: Bipartite matching

Let $G = (U, V, E)$ a bipartite graph, i.e. $E \subset U \times V$.
A matching $M$ is a set of edges so that each node of $U$ or $V$ is matched only once in $M$.
We want a matching of maximum cardinality.



# Application: Assignment problem

Assigning worker $i \in [N]$ to job $j \in [M]$ has cost $W_{ij}$. We want to minimize costs.

## Min-cost (or max-weight) bipartite matching

Find, in a weighted bipartite graph, a matching of given size, in which the sum of weights on the edges is minimum (or maximum).

$$\min_{A \in \{0,1\}^{N \times M}} \sum_{i=1}^{N} \sum_{j=1}^{M} A_{ij} W_{ij} \quad A\mathbf{1}_M \in \{0,1\}^N \quad \mathbf{1}_N A \in \{0,1\}^M$$

# Application: Assignment problem

Assigning worker $i \in [N]$ to job $j \in [M]$ has cost $W_{ij}$. We want to minimize costs.

## Min-cost (or max-weight) bipartite matching

Find, in a weighted bipartite graph, a matching of given size, in which the sum of weights on the edges is minimum (or maximum).

$$\min_{A \in \{0,1\}^{N \times M}} \sum_{i=1}^{N} \sum_{j=1}^{M} A_{ij} W_{ij} \quad A\mathbf{1}_M \in \{0,1\}^N \quad \mathbf{1}_N A \in \{0,1\}^M$$

Dual Hungarian algorithm $O(V^4)$ or $O(V^3)$ according to optimization

Primal Min-cost max-flow $O(V^4)$ if Bellman-Ford or $O(V^3)$ if Dijkstra ($> 0$) but one can always have nonnegative edges (Schrijver et al., 2003)

$O(VE \log V)$ if Dijkstra with a sparse adjacency matrix.

- Erickson, Jeff (2019). *Algorithms*. Jeff Erickson. URL: https://algorithms.wtf.
- Schrijver, Alexander et al. (2003). *Combinatorial optimization: polyhedra and efficiency*. Vol. 24. Springer.