Lesson 3

Cuts, flows and graphical models

Carlo Mannino
SINTEF Digital and University of Oslo

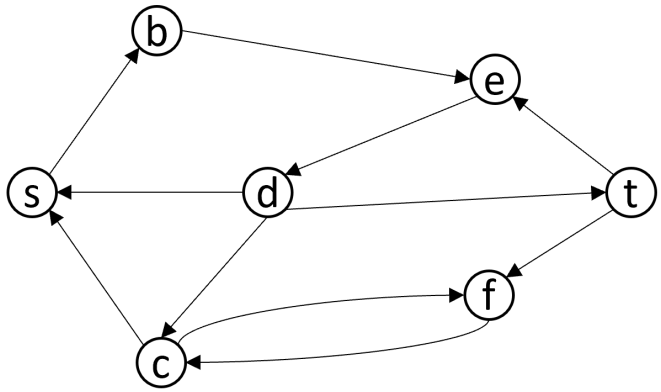
Outline

1. Max-flow and min-cut
2. Graphical Model and Energy Minimization

Directed Graphs

Schrijver, A., 2003. Combinatorial optimization: polyhedra and efficiency (Vol. 24). Berlin: Springer.

- *Directed graph* (or *digraph*) is a pair $G = (V, A)$
- V (finite) set of *nodes* (or *vertices*)
- A family of ordered pairs from V, called *arcs* (*directed edges*)
- We let $n = |V|$ and $m = |A|$



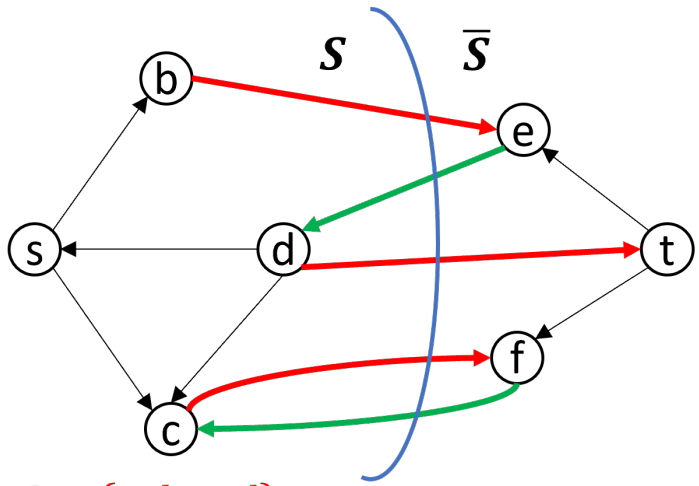
$V = (s, b, c, d, e, f, t)$
$A = \{(s, b), (b, e), (c, s), (c, f), (d, c),$
$(d, s)\ (d, t), (e, d), (f, c), (t, f)\}$

- Directed Path: sequence $P = (v_0, a_1, v_1, \ldots, a_k, v_k)$, $v_0, \ldots, v_k \in V$ and distinct, $a_i = (v_{i-1}, v_i)$, $i = 1, \ldots, k$
- Directed Cycle: sequence $C = (v_1, a_1, \ldots, v_k, a_k)$, $v_1, \ldots, v_k \in V$ and distinct, $a_i = (v_i, v_{i+1})$, $i = 1, \ldots, k-1$, $a_k = (v_k, v_1)$

st-cut

- *Let* $S \subseteq V$ and let $\overline{S} = V \setminus S$.
- $\delta^{out}(S) = \{(u, v) : u \in S, v \in \overline{S}\}$: set of arcs outgoing from S.
- $\delta^{in}(S) = \{(v, u) : v \in \overline{S}, u \in S\}$: set of arcs coming into S.

- We write $\delta^{out}(\{u\}) = \delta^{out}(u) = \delta^{+}(u)$ the *outgoing star* of $u \in V$.
- $\delta^{in}(\{u\}) = \delta^{in}(u) = \delta^{-}(u)$ the *incoming star* of $u \in V$.
- $s, t \in V$ and $s \in S$, $t \in \overline{S}$, we call $\delta^{out}(S)$ an *st*-cut.

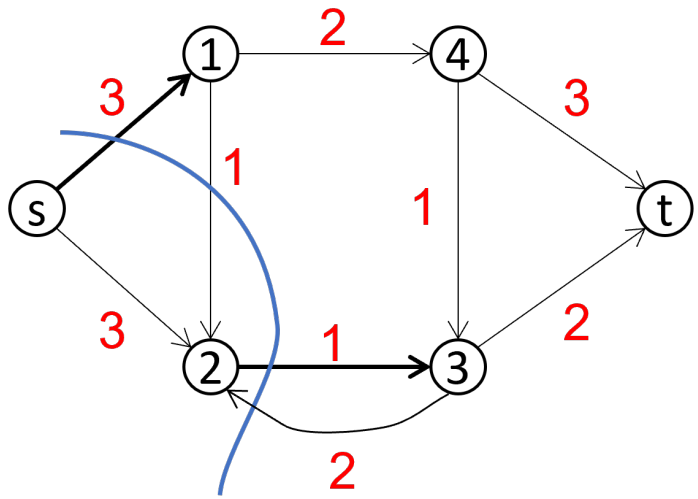


$S = \{a, b, c, d\}$
$\delta^{out}(S) = \{(b, e), (c, f), (d, t)\}$
$\delta^{in}(S) = \{(e, d), (f, c)\}$

Capacity of s-t cuts

- Let $c : A \to \mathbb{R}_+$ be the *capacity* function ($c_a = c(a)$ capacity of $a \in A$).
- The capacity of the st-cut $\delta^{out}(S)$ is the quantity $c(\delta^{out}(S)) = \sum_{e \in \delta^{out}(S)} c(e)$.

- In the figure, the capacities of the arcs are the red numbers.
- The capacity of the st-cut $\delta^{out}(\{s, 2\})$ is 6.



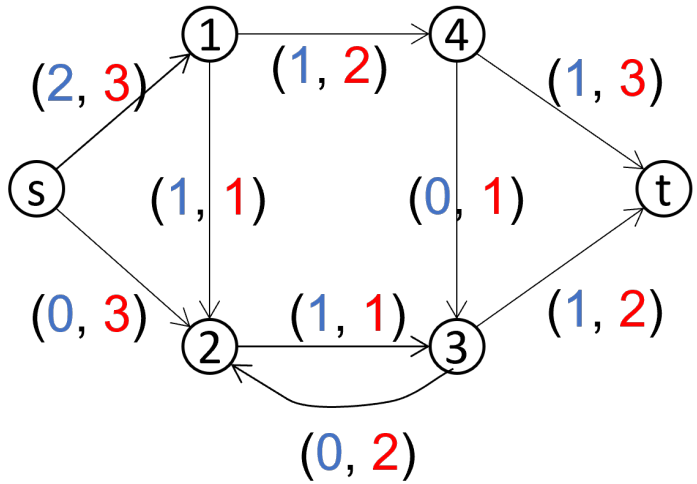
s-t flow

- Given a directed graph $G = (V, A)$, with $s, t \in V$, $s \neq t$, and capacity function c
- An $s - t$ flow of G under c is a function $f : A \to \mathbb{R}_+$ such that:

$$\begin{array}{lll} (i) & f(a) \geq 0 & a \in A \\ (ii) & f(\delta^{out}(v)) = f(\delta^{in}(v)) & v \in V \setminus \{s, t\} \\ (iii) & f(a) \leq c(a) & a \in A \end{array} \tag{1}$$

- $f(a)$ is called *flow on arc a*.
- (ii) is the flow conservation constraint at intermediate nodes
- (iii) is the capacity constraint

- In the figure: (f, c), f is the flow on the arc, c the capacity of the arc



Value of the flow

- The *value of a flow f* is the net flow out from *s*:

$$\text{value}(f) := f(\delta^{out}(s)) - f(\delta^{in}(s))$$

Definition

A *maximum flow* is an *s*-*t* flow under *c* of maximum value. The *maximum flow problem* is to find a maximum flow.

- For a flow $f : A \to \mathbb{R}$, the *netflow function* is the function $\mathit{netflow}_f : \mathcal{P}(V) \to \mathbb{R}$ defined by

$$\mathit{netflow}_f(U) := f(\delta^{out}(U)) - f(\delta^{in}(U))$$

  where $\mathcal{P}(v)$ denotes the power set of V.
- Note that $\text{value}(f) = \mathit{netflow}_f(s)$
- Note that, for $v \neq s, t$, $\mathit{netflow}_f(v) = 0$.

Net flow Lemma

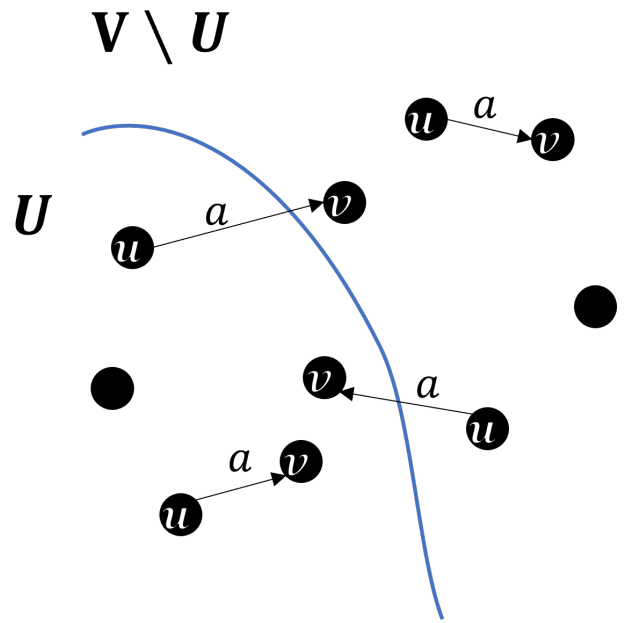
Lemma

Let $G = (V, A)$ be a digraph, $f : A \to \mathbb{R}$, and $U \subseteq V$. Then:

$$netflow_f(U) := \sum_{v \in U} netflow_f(v)$$

- Count the molteplicity of each $a = (u, v) \in A$ in both sides.
- if $u \in U$, $v \in V \setminus U$, $f(a)$ appears both in l.h.s. and in r.h.s.
- if $u \in V \setminus U$, $v \in U$, $-f(a)$ appears both in l.h.s. and in r.h.s.
- if $u, v \in V \setminus U$, neither $f(a)$ nor $-f(a)$ appear in l.h.s. or in r.h.s.
- if $u, v \in U$, both $f(a)$ and $-f(a)$ appear only in r.h.s. and cancel out.

Four cases for $a = (u, v)$



Net flow through s-t cut

Lemma

Let f be a flow under capacity c. Let $U \subset V$, $s \in U$, $t \in V \setminus U$. Then

$$value(f) \leq c(\delta^{out}(U))$$

Since f is a flow, $netflow_f(v) = 0$ for $v \in U \setminus \{s\}$.

$\rightarrow$ $netflow_f(U) = netflow_f(s) :=$value$(f)$.

$\rightarrow$ value$(f) = netflow_f(U) = \sum_{a \in \delta^{out}(U)} f(a) - \sum_{a \in \delta^{in}(U)} f(a) \leq \sum_{a \in \delta^{out}(U)} f(a)$

$\rightarrow$ value$(f) \leq \sum_{a \in \delta^{out}(U)} f(a) \leq \sum_{a \in \delta^{out}(U)} c(a) := c(\delta^{out}(U))$ $\square$.

In other words:

The value of a maximum s-t flow is at most the capacity of any s-t cut.

The beautiful max-flow min-cut theorem

- We are now able to state, and then to prove, one of the central theorems in combinatorial optimization
- From the previous lemma, we know that the capacity of a minimum capacity *s*-*t* cut is an upper bound on the value of an *s*-*t* flow.
- Next theorem shows that a stronger result holds

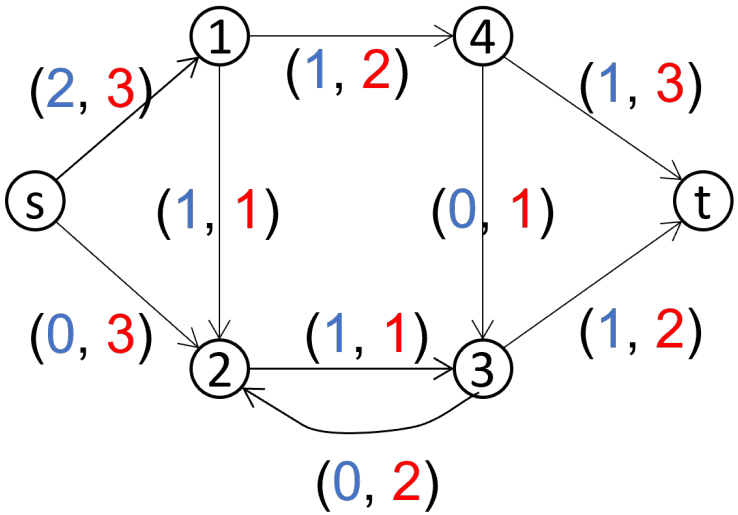
Theorem (The max-flow min-cut theorem)

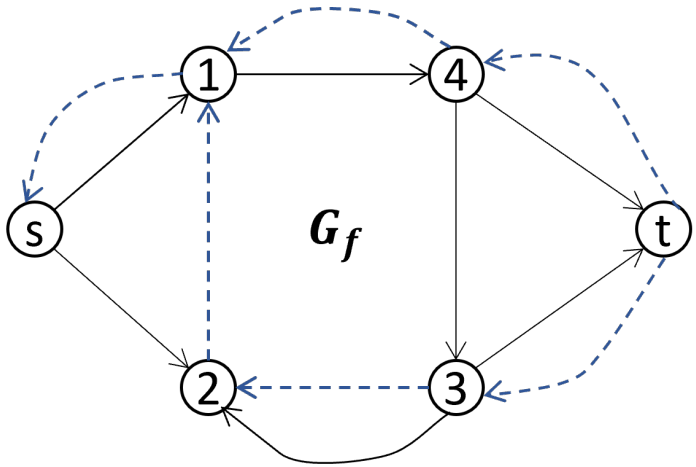
Let $G = (V, A)$ be a digraph, with $s, t \in V$, $s \neq t$, and let $c : A \to \mathbb{R}$ be a capacity function. Then the value of a maximum s-t flow flow under c equals the value of a minimum capacity s-t cut of G.

- The max-flow min-cut theorem is a *strong duality* result
- To prove it we introduce the *residual graph*

Residual graph

- Given graph $G = (V, A)$, capacity function $c : A \to \mathbb{R}$, and flow $f : A \to \mathbb{R}$, $0 \leq f \leq c$.
- For each $a = (u, v) \in A$ define $a^{-1} = (v, u)$, and let $A^{-1} = \{a^{-1} : a \in A\}$ (*backward arcs*)
- Define $A_f = \{a \in A : f(a) < c(a)\} \cup \{a^{-1} \in A^{-1} : f(a) > 0\}$. $A_f \cap A$ *forward arcs*.
- $G_f = (V, A_f)$ is called the *residual graph* of f.

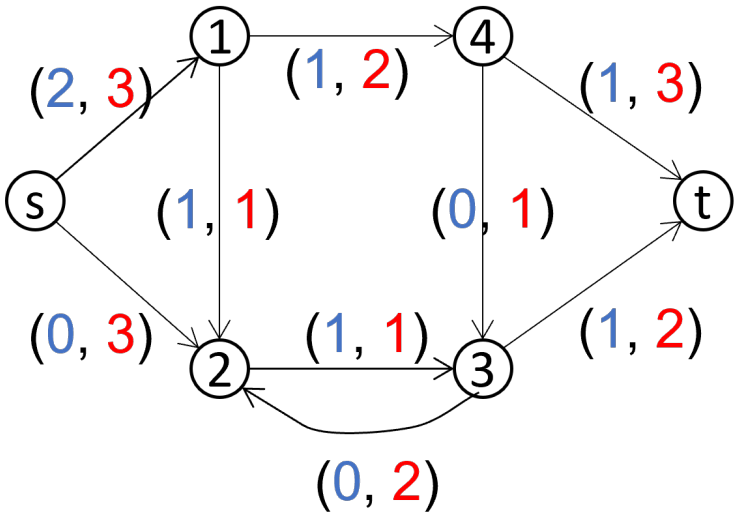


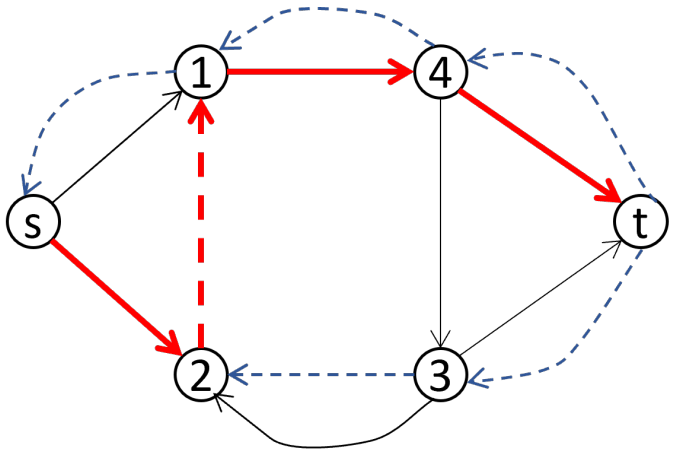


Augmenting path

Lemma (Augmenting Path lemma)

Let f be an s-t flow under c. If there is a directed path P_f from s to t in G_f, then f is not maximum





$P_f = (s, (s, 2), 2, (2,1), 1, (1,4)4, (4, t), t)$

- P_f is called an *augmenting path*

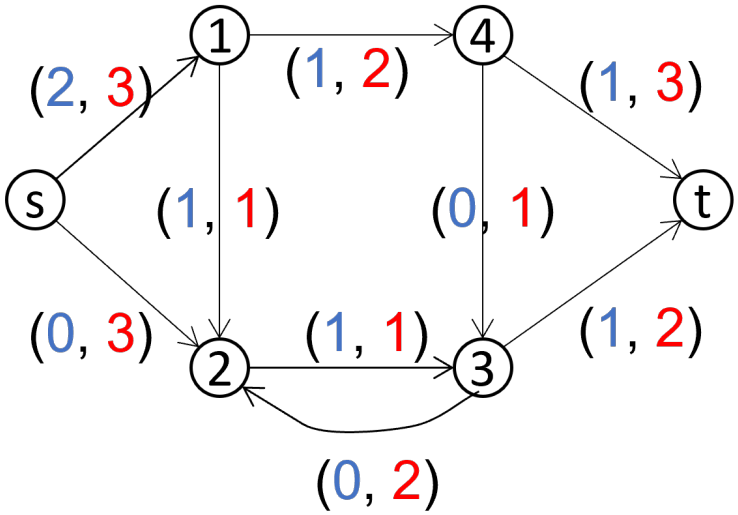
Proof Define a suitably small $\epsilon > 0$

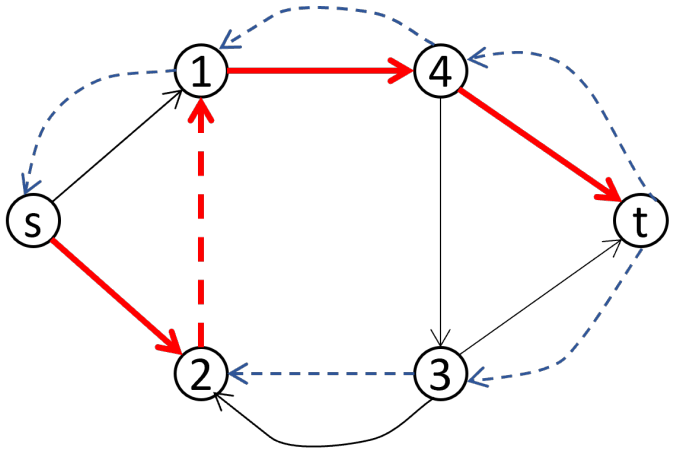
- We **increase** the flow by ϵ on every **forward arc** of P_f
- We **decrease** the flow by ϵ on every **backward arc** of P_f
- Formally, for $a \in A$, define $\chi^{P_f} \in \mathbb{R}^A$ by

$$\chi^P(a) = \begin{cases} 1 & \text{if } P_f \text{ traverses } a \\ -1 & \text{if } P_f \text{ traverses } a^{-1} \\ 0 & \text{if } P_f \text{ traverses neither } a \text{ nor } a^{-1} \end{cases}$$

- Let $f' := f + \epsilon\chi^{P_f}$.
- We show that for ϵ small enough f' is a feasible flow.
- Moreover, value(f') > value(f).

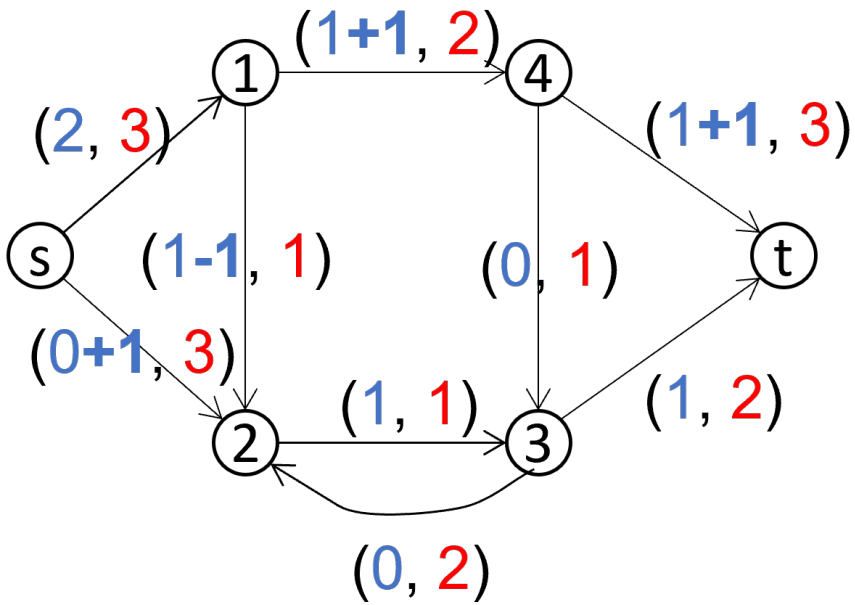
Example





$P_f = (s, (s, 2), 2, (2,1), 1, (1,4)4, (4, t), t)$

- We can let $\epsilon = 1$.
- The flow increases by 1 on $(s, 2), (1, 4), (4, t)$
- The flow decreases by 1 on $(1, 2)$

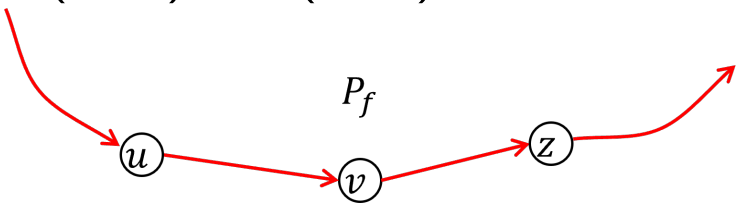


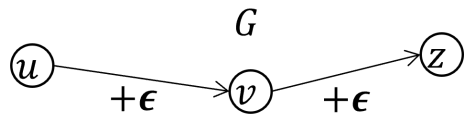
Proof of augmenting path lemma

- Let $f' := f + \epsilon\chi^{P_f}$, $\epsilon > 0$. For ϵ small enough f' is a feasible flow.

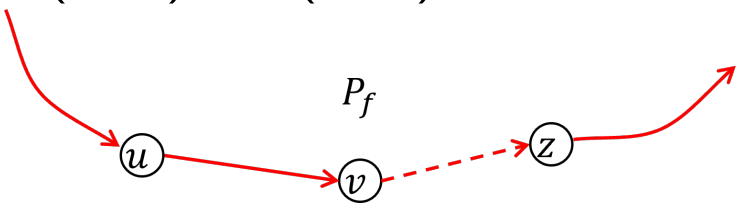
  - **Flow conservation constraints at intermediate node** v.
    Assume P_f traverses (u, v) and (v, z). We have 4 cases:

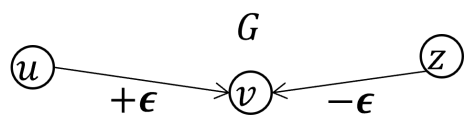
1.) $(u, v) \in A$ and $(v, z) \in A$. Then $f'(u, v) = f(u, v) + \epsilon$ and $f'(v, z) = f(v, z) + \epsilon$. Inflow and outflow in v increase both by ϵ.





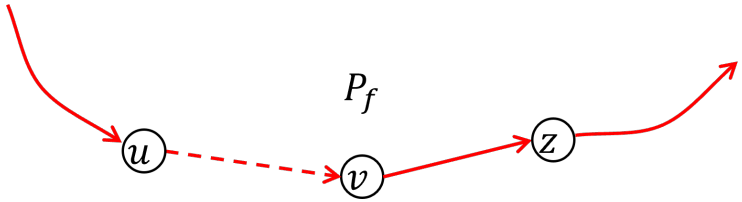
2.) $(u, v) \in A$ and $(v, z) \in A^{-1}$. Then $f'(u, v) = f(u, v) + \epsilon$ and $f'(z, v) = f(z, v) - \epsilon$. Inflow in v increases and decreases by ϵ.

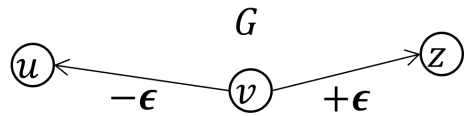




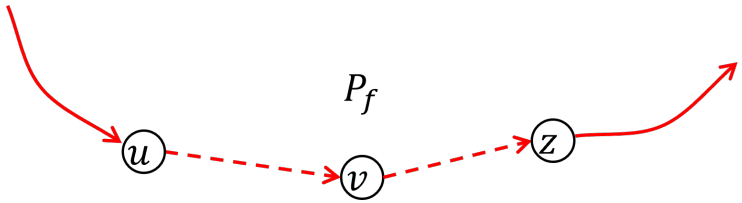
Proof of augmenting path lemma

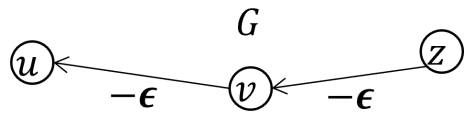
3.) $(u, v) \in A^{-1}$ and $(v, z) \in A$. Then $f'(v, u) = f(v, u) - \epsilon$ and $f'(v, z) = f(v, z) + \epsilon$. Outflow increases and decreases by ϵ.





4.) $(u, v) \in A^{-1}$ and $(v, z) \in A^{-1}$. Then $f'(v, u) = f(v, u) - \epsilon$ and $f'(z, v) = f(z, v) - \epsilon$. Both inflow and outflow in v decrease by ϵ.





Exercise

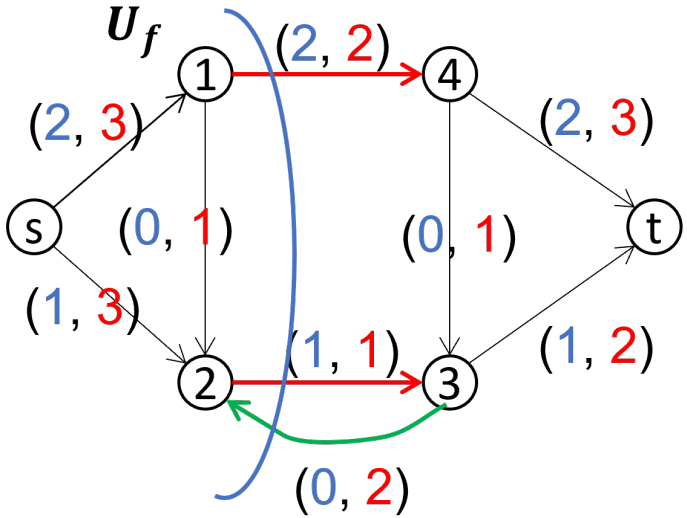
1. Find $\epsilon > 0$ such that non-negativity and capacity constraints on the arc flow are satisfied.
2. Show that the flow outgoing s increases by ϵ and thus value(f') =value$(f) + \epsilon$.

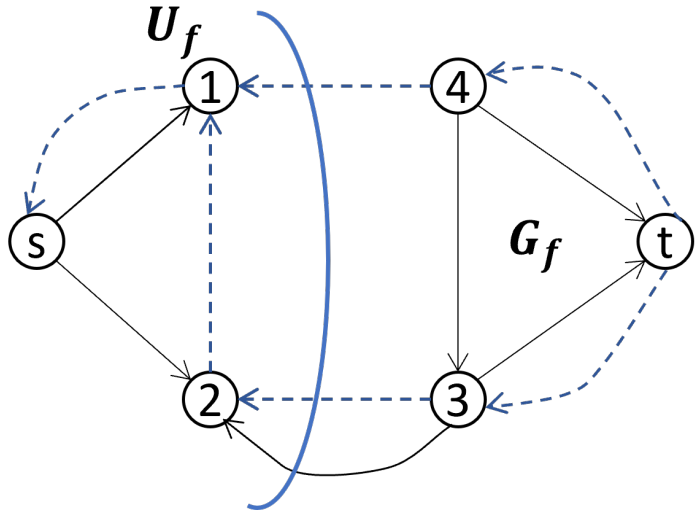
□

Strong duality

Lemma (Strong duality lemma)

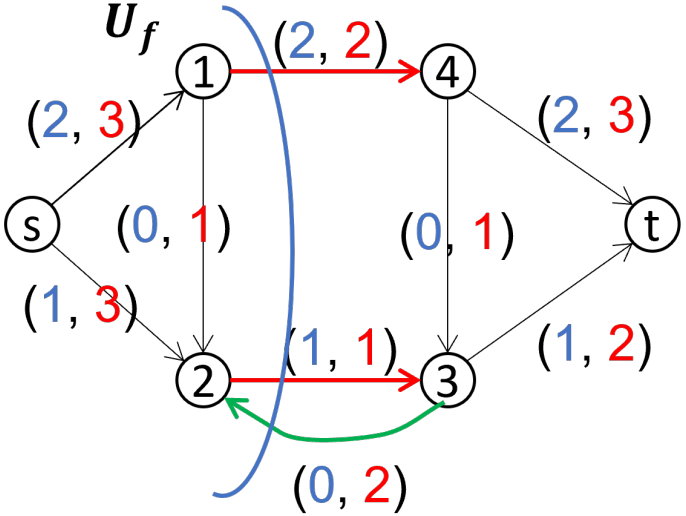
Let f be an s-t flow of $G = (V, A)$ under capacity c. If there is no directed path P_f from s to t in G_f, then there exists an s-t cut $\delta^{out}(U_f)$ of G such that $c(\delta^{out}(U_f)) = value(f)$ and f is a maximum flow.

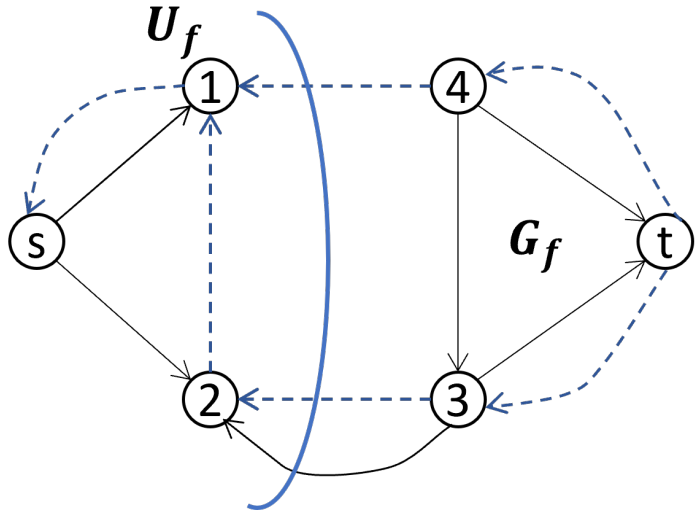




- Let $U_f = \{u \in V$: there is a path from s to u in $G_f\}$.
- Since $t \notin U_f$, $\delta^{out}(U_f)$ is an s-t cut of G.
- We show that $c(\delta^{out}(U_f))$ =value(f).

Strong duality





- If $a = (u, v) \in \delta^{out}(U_f)$, then $u \in U_f$, $v \notin U_f$ implies $a \notin A_f$. This implies $f(a) = c(a)$.
- If $a = (u, v) \in \delta^{in}(U_f)$, then $v \in U_f$, $u \notin U_f$ implies $a^- = (v, u) \notin A_f$. This implies $f(a) = 0$.
- Then value$(f) = netflow_f(U_f) = \sum_{a \in \delta^{out}(U_f)} f(a) - \sum_{a \in \delta^{in}(U_f)} f(a) = \sum_{a \in \delta^{out}(U_f)} c(a) = c(\delta^{out}(U_f))$.
- Since value$(f) \leq c(\delta^{out}(U))$ for any s-t cut $\delta^{out}(U)$, then f is a maximum flow and $\delta^{out}(U_f)$ a minimum capacity s-t cut.

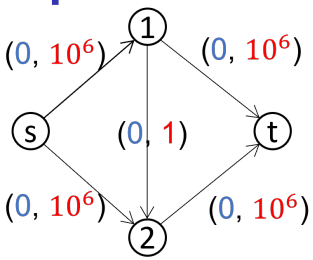
An algorithm for the maximum flow problem

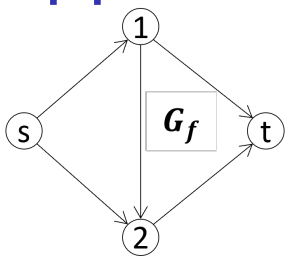
A simple algorithm for the maximum flow problem.

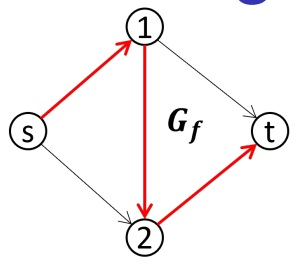
1. Begin with the empty flow $f = \mathbf{0}$
2. Construct the residual network G_f.
3. If there is no $s - t$ path in G_f, STOP (f is a maximum flow).
4. Otherwise choose an $s - t$ path P and $\epsilon > 0$ such that $f + \epsilon\chi^P$ is a feasible st-flow.
5. Let $f := f + \epsilon\chi^P$ and GOTO Step 2.

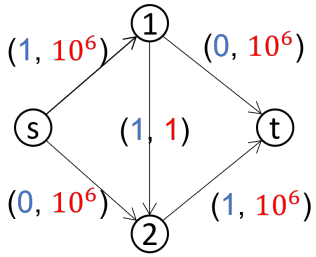
- Under some mild conditions (e.g. c rational), the algorithm always converges.
- There exist algorithms that solve the problem in strongly polynomial time

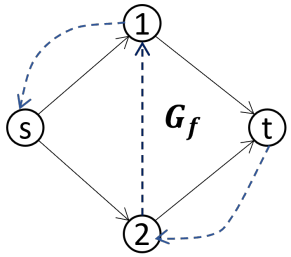
A pathological application of the augmenting algorithm

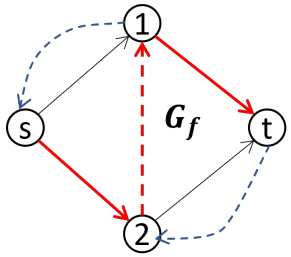


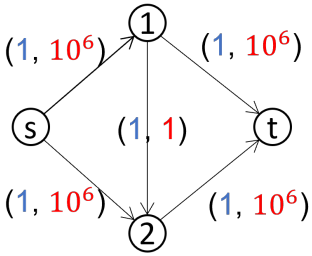


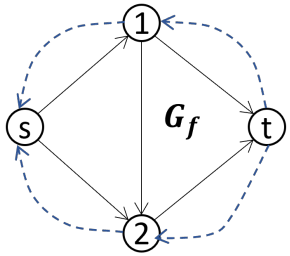


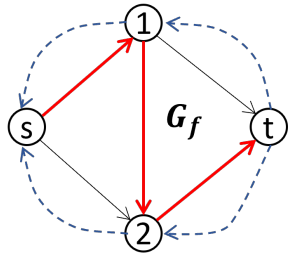












- The flow value increases by 1 unit at a time
- If finding an $s-t$ path requires $\mathcal{O}(m)$ time and c_{max} is the maximum capacity, you may need up to $\mathcal{O}(c_{max} \cdot m)$ time.
- Is $\mathcal{O}(c_{max} \cdot m)$ time polynomial time?

Flows and LPs

- The Maximum Flow Problem is an LP problem:

$$\begin{array}{ll} \max & \sum_{su \in \delta^{out}(s)} f_{su} - \sum_{us \in \delta^{in}(s)} f_{us} \\ & \sum_{uv \in \delta^{out}(u)} f_{uv} - \sum_{vu \in \delta^{in}(u)} f_{vu} = 0 \qquad u \in V \setminus \{s, t\} \\ & 0 \leq f_{uv} \leq c_{uv} \qquad uv \in A \\ & f \in \mathbb{R}^A \end{array}$$

Lemma

If $c \in Z_+^A$, i.e. the capacities are non-negative integers, then there exists an optimal integer flow vector.

Exercise

Prove the lemma!

The matching problem in bipartite graphs

Exercise

A *bipartite graph* is an undirected graph $G = (V, E)$, where V can be partitioned into two sets U, W such that every edge has one extreme in U and the other extreme in W. A *matching* $M \subseteq E$ is a set of edges such that no two edges in M share a node. Show that the problem of finding a maximum cardinality matching in a graph can be transformed into the problem of finding a maximum $s - t$ flow in a suitable digraph under some capacity c.

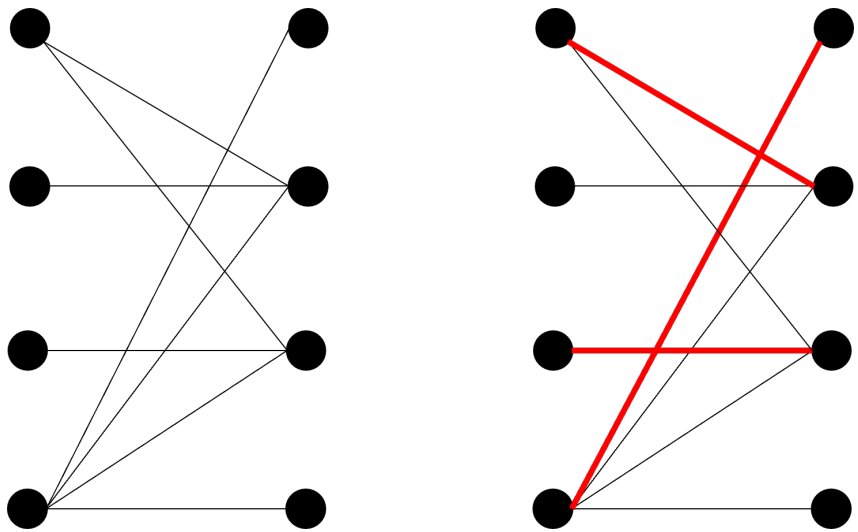


Figure: A bipartite graph and a maximum cardinality matching (in red)

Graphical Model

Savchynskyy, B., 2019. Discrete graphical models - an optimization perspective. *Foundations and Trends in Computer Graphics and Vision*, 11(3-4), pp.160-429.

- $G = (V, A)$ directed graph
- Finite set of labels $\mathcal{Y}_u$ associated with each $u \in V$.
- Cost function $\theta_u : \mathcal{Y}_u \to \mathbb{R}$: $\theta_u(l)$ cost assigning label $l \in \mathcal{Y}_u$ to u.
- Pairwise cost function $\theta_{uv} : \mathcal{Y}_u \times \mathcal{Y}_v \to \mathbb{R}$, for all $(u, v) \in E$: $\theta_{uv}(l, m)$ is the cost of assigning label $l \in \mathcal{Y}_u$ to u and label $m \in \mathcal{Y}_v$ to v.
- Set of all labelings $\mathcal{Y}_V = \prod_{u \in V} \mathcal{Y}_u$

Definition (Graphical Model)

A *Graphical Model* is a triple $(G, \mathcal{Y}_V, \theta)$

Energy minimization

Definition (Energy minimization problem)

Given graphical model $(G, \mathcal{Y}_V, \theta)$, the *energy minimization* problem or *Maximum a Posteriori (MAP)* inference problem is the problem of finding a labelling y^* of minimum total cost:

$$y^* = arg \min_{y \in \mathcal{Y}_V} \left[E(y, \theta) := \sum_{u \in V} \theta_u(y_u) + \sum_{(u,v) \in A} \theta_{uv}(y_u, y_v) \right] \tag{2}$$

- Probabilistic interpretation: we are given an exponential probability distribution associated with $E(y, \theta)$:

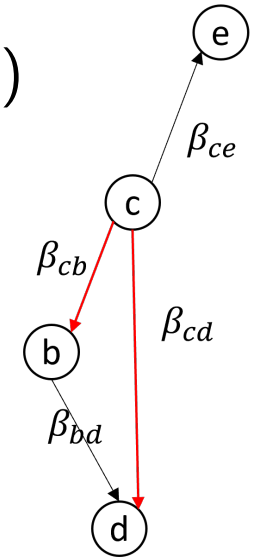
$$p(y) = \frac{1}{Z(\theta)} \exp(-E(y, \theta))$$

- y^* in (2) is the *most probable* labelling.

Example: Optimal recovery policies

Benth, F.E., Dahl, G. and Mannino, C., 2012. Computing optimal recovery policies for financial markets.*Operations research*, 60(6), pp.1373-1388.

- Evaluates the default risk of each company using a Markov random field model
- Directed graph $G = (V, A)$: V are the companies, edge (u, v) describes how default risk transmits from u to v
- Nodes may represent entire sectors of the economy or even the overall state of the economy.
- For $uv \in A$, $\beta_{uv} \geq 0$ indicates how dependent v is on u
- At time t_0 (now) we have certain information and we predict what (most likely) will happen at time t_1 (e.g., in a month)
- For $u \in V$, $\mathcal{Y}_u = \{0, 1\}$. If $y_u = 1$, u will default (at time t_1); if $y_u = 0$, u will not default.
- Vector y is the *default state vector*.



Example: Optimal recovery policies

- Prior probability distribution of y (*Ising model*):

$$\pi(y) = \frac{1}{s} \exp\left\{ \sum_{v \in V} [a_v^1 y_v + a_v^0 (1 - y_v)] + \sum_{(u,v) \in A} \beta_{uv} I(y_u \leq y_v) \right\}.$$

- We want to find $y^* = arg \max \pi(y), y \in \{0,1\}^V$
- The *indicator function* $I(y_u \leq y_v)$ equals 1 if $y_u \leq y_v$, 0 otherwise
- s normalization constant (making $\pi(y)$ a proper probability)
- α_v^1, α_v^0 reflect a priori information on financial position of agent v
- i.e. $(1/s)e^{\alpha_v^1}$ (resp. $(1/s)e^{\alpha_v^0}$) is an a priori probability that v defaults (resp. survives).
- We may assume $\alpha_v^1, \alpha_v^0 \leq 0$ (show that subtracting a constant b from both α_v^1, α_v^0 does not change the optimal vector(s) y^*).

Example: Optimal recovery policies

- We can equivalently maximize the logarithm $L(y)$ of $\pi(y)$:

$$L(y) = \sum_{v \in V} [a_v^1 y_v + a_v^0 (1 - y_v)] + \sum_{(u,v) \in A} \beta_{uv} I(y_u \leq y_v) \qquad (3)$$

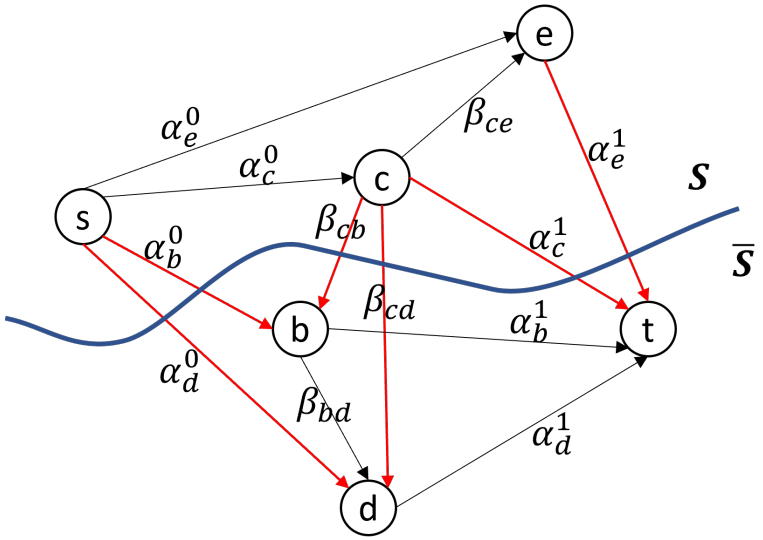
- We introduce an auxiliary graph $G' = (V', A')$. $V' = V \cup \{s, t\}$.
- $A' = A \cup A_s \cup A_t$, with $A_s = \{(s, v) : v \in V\}$, $A_t = \{(v, t) : v \in V\}$.
- Weights: $w_{uv} = -\beta_{uv}$, $(u, v) \in A$. $w_{sv} = \alpha_v^0$, $w_{vt} = \alpha_v^1$, $v \in V$

Lemma

Let $c_e = -w_e$, for $e \in A'$. The maximum value $L(y^*)$ to (3) equals the capacity of a minimum capacity st-cut $\delta^{out}(S^*)$.

Proof of Lemma 10

- Consider $y \in \{0,1\}^V$ and let $S_y = \{u \in V : y_u = 1\}$. $\overline{S_y} = \{u \in V : y_u = 0\}$
- Note that
  - $\delta^{out}(\{s\} \cup S_y)$ is an st-cut in G'
  - $\sum_{(u,v)\in E} \beta_{uv} = \beta^*$ (constant)



- $w(\delta^{out}(\{s\} \cup S_y)) = \sum_{u\in \bar{S}_y} \alpha_u^0 + \sum_{u\in S_y} \alpha_u^1 - \sum_{u\in S_y, v\in \bar{S}_y} \beta_{uv} = \sum_{u\in V: y_u=0} \alpha_u^0 + \sum_{u\in V: y_u=1} \alpha_u^1 - \sum_{u\in S_y, v\in \bar{S}_y} \beta_{uv}$.
- Since $\sum_{(u,v)\in E} \beta_{uv} = \sum_{y_u \le y_v : (u,v)\in E} \beta_{uv} + \sum_{y_u > y_v : (u,v)\in E} \beta_{uv}$ then
- $\sum_{y_u > y_v : (u,v)\in E} \beta_{uv} = \beta^* - \sum_{y_u \le y_v : (u,v)\in E} \beta_{uv}$ and thus
- $w(\delta^{out}(\{s\} \cup S_y)) = \sum_{u\in V: y_u=0} \alpha_u^0 + \sum_{u\in V: y_u=1} \alpha_u^1 + \sum_{y_u \le y_v : (u,v)\in E} \beta_{uv} - \beta^* = L(y) - \beta^*$

□

References

- Ahuja, R.K., Magnanti, T.L. and Orlin, J.B., 1993. Network flows. Prentice Hall
- Benth, F.E., Dahl, G. and Mannino, C., 2012. Computing optimal recovery policies for financial markets. *Operations research*, 60(6), pp.1373-1388.
- Savchynskyy, B., 2019. Discrete graphical models - an optimization perspective. *Foundations and Trends in Computer Graphics and Vision*, 11(3-4), pp.160-429.
- Schrijver, A., 2003. Combinatorial optimization: polyhedra and efficiency (Vol. 24). Berlin: Springer.