Combinatorial Optimization and Graph Theory
ORCO
Introduction + Flows

Zoltán Szigeti

Teachers

Teachers

1. SZIGETI, Zoltán (8 weeks)
   - Professor at Ensimag, e-mail: zoltan.szigeti@grenoble-inp.fr
2. STEHLIK, Matej (4 weeks)
   - Assistant Professor at UGA, e-mail: matej.stehlik@grenoble-inp.fr

Researchers:

1. Research at G-SCOP Laboratory.
2. Research subjects:
   - Combinatorial Optimization,
   - Graph Theory,
   - Connectivity,
   - Matchings,
   - Matroids.

Course:

Combinatorial Optimization

1. Discrete optimization part of Operations Research, consists of "Finding the best solution in a very large set of possibilities".
   - Previously seen:
     - Shortest paths,
     - Minimum cost spanning trees.
2. Structural results
   - Previously seen:
     - Subpath of a shortest path is a shortest path.
     - Maximal forest is maximum forest.
3. Efficient algorithms
   - Previously seen:
     - Bellmann, Dijkstra, Floyd-Warshall for shortest paths,
     - Kruskal (greedy) for minimum cost spanning trees.

Planning

Subjects treated in my part:

1. Network flows,
2. Push-Relabel algorithm for flows,
3. Matchings in bipartite graphs,
4. Matchings in general graphs,
5. Matroids,
6. Submodular functions in graph theory,
7. Paper presentations (2 weeks),

Citation about flows :

"*But anyone who has experienced flow knows that the deep enjoyment it provides requires an equal degree of disciplined concentration.*"

Mihály Csikszentmihályi

Books

Books for further study

1. **Ahuja, Magnanti, Orlin,** *Network flows; Theory, Algorithms and Applications,*
2. **Cook, Cunningham, Pulleyblank, Schrijver,** *Combinatorial Optimization,*
3. **Frank,** *Connections in Combinatorial Optimization,*
4. **Korte, Vygen,** *Combinatorial Optimization; Theory and Algorithms,*
5. **Lovász, Plummer,** *Matching Theory,*
6. **Schrijver,** *Combinatorial Optimization; Polyhedra and Efficiency,* 3 volumes.

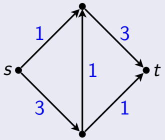
Introduction to flows

Problem

How many trucks can we send from a starting point to a destination point respecting the capacity constraints of the streets?

Model

1. Given
   1. a directed graph $G = (V, A)$,
   2. source $s \in V$ and sink $t \in V$,
   3. a capacity function g on the arcs,



2. find a set $\mathcal{P}$ of (s, t)-paths such that each arc e belongs to at most $g(e)$ paths of $\mathcal{P}$.
3. It suffices to know the number $x(e)$ of paths in $\mathcal{P}$ containing $e \in A$.
4. The function $x : A \to \mathbb{R}$ is called flow.

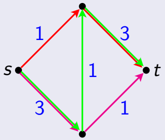
Introduction to flows

Problem

How many trucks can we send from a starting point to a destination point respecting the capacity constraints of the streets?

Model

1. Given
   1. a directed graph $G = (V, A)$,
   2. source $s \in V$ and sink $t \in V$,
   3. a capacity function g on the arcs,



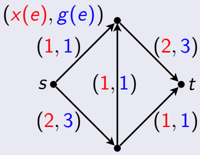
2. find a set $\mathcal{P}$ of (s, t)-paths such that each arc e belongs to at most $g(e)$ paths of $\mathcal{P}$.
3. It suffices to know the number $x(e)$ of paths in $\mathcal{P}$ containing $e \in A$.
4. The function $x : A \to \mathbb{R}$ is called flow.

Problem

How many trucks can we send from a starting point to a destination point respecting the capacity constraints of the streets?

Model

1. Given
   1. a directed graph $G = (V, A)$,
   2. source $s \in V$ and sink $t \in V$,
   3. a capacity function g on the arcs,

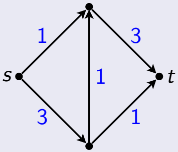


2. find a set $\mathcal{P}$ of (s,t)-paths such that each arc e belongs to at most $g(e)$ paths of $\mathcal{P}$.
3. It suffices to know the number $x(e)$ of paths in $\mathcal{P}$ containing $e \in A$.
4. The function $x : A \to \mathbb{R}$ is called flow.

Definition of flows

Definition

1. Given
   1. a directed graph $G = (V, A)$,
   2. $s, t \in V$ such that $\delta^-(s) = \emptyset = \delta^+(t)$,
   3. a non-negative capacity g on the arcs,



2. a function x on the arcs is
   1. an **(s,t)-flow** if the **flow conservation** is satisfied:

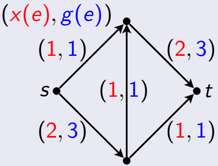
$$\sum_{uv \in A} x(uv) = \sum_{vu \in A} x(vu) \quad \forall v \in V \setminus \{s, t\}.$$

   2. **feasible** if the **capacity contraint** is satisfied:

$$0 \leq x(e) \leq g(e) \quad \forall e \in A.$$

Definition

1. Given
   1. a directed graph $G = (V, A)$,
   2. $s, t \in V$ such that $\delta^-(s) = \emptyset = \delta^+(t)$,
   3. a non-negative capacity g on the arcs,



2. a function x on the arcs is
   1. an **(s,t)-flow** if the **flow conservation** is satisfied:

$$\sum_{uv \in A} x(uv) = \sum_{vu \in A} x(vu) \quad \forall v \in V \setminus \{s, t\}.$$

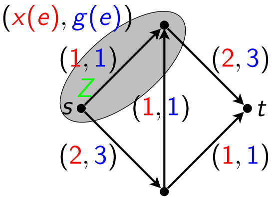
   2. **feasible** if the **capacity contraint** is satisfied:

$$0 \le x(e) \le g(e) \quad \forall e \in A.$$

Notation

Given directed graph $G = (V, A), s, t \in V$, capacity g, flow x, $Z \subseteq V$,

1. $\delta^+(Z)$: the arcs leaving Z,
2. **Out-value of** Z: $d_x^+(Z) := \sum_{e \in \delta^+(Z)} x(e)$,
3. **Flow conservation**: $d_x^-(v) = d_x^+(v)$,
4. **Flow value**: $val(x) := d_x^+(s)$,
5. (s,t)**-cut** Z: if $s \in Z \subseteq V \setminus t$,
6. **Capacity of** (s,t)**-cut** Z: $cap(Z) := d_g^+(Z)$.



$d_x^+(Z) = 4$
$d_x^+(s) = 3$
$d_g^+(Z) = 6$

Flow value

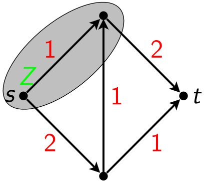
Lemma

For all (s,t)-flow x and for all (s,t)-cut Z:

$$\text{val}(x) = d_x^+(Z) - d_x^-(Z).$$

Proof

$$\begin{aligned}
val(x) &= d_x^+(s) \\
&= d_x^+(s) - \overbrace{d_x^-(s)}^{0} + \sum_{v \in Z-s} \overbrace{(d_x^+(v) - d_x^-(v))}^{0} \\
&= \sum_{v \in Z} (d_x^+(v) - d_x^-(v)) \\
&= d_x^+(Z) - d_x^-(Z).
\end{aligned}$$



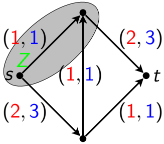
Max Flow $\leq$ Min Cut

Lemma

For all g-feasible (s,t)-flow x and for all (s,t)-cut Z: $\mathrm{val}(x) \leq \mathrm{cap}(Z)$.

Proof

$$\begin{aligned} val(x) &= d_x^+(Z) - d_x^-(Z) \\ &\leq d_g^+(Z) - d_0^-(Z) \\ &= d_g^+(Z) = cap(Z). \end{aligned}$$



Remark

If x is a g-feasible (s,t)-flow and Z is an (s,t)-cut such that $\mathrm{val}(x) = \mathrm{cap}(Z)$, then they are optimal.

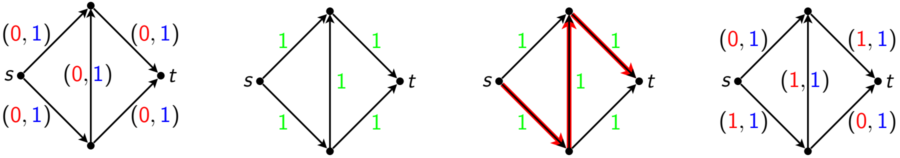
Problem: How to find

1. a g-feasible (s,t)-flow of maximum value and
2. an (s,t)-cut of minimum capacity?

Flow augmentation

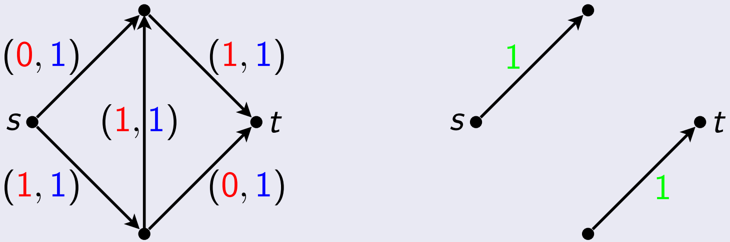
First ideas

1. $x(e) = 0 \ \forall e \in A$ is a feasible flow.
2. How to augment a flow?
3. $G' := (V, A^1_x)$ where $A^1_x := \{uv \in A : x(uv) < g(uv)\}$.
4. If there exists an (s,t)-path P in G' then we can augment the value of the flow by $\varepsilon^1_x := \min\{g(uv) - x(uv) : uv \in A(P) \cap A^1_x\}$,
5. $x'(uv) := \begin{cases} x(uv) + \varepsilon^1_x & \text{if } uv \in A(P) \cap A^1_x \\ x(uv) & \text{otherwise.} \end{cases}$



Flow augmentation

Example: this idea is not enough

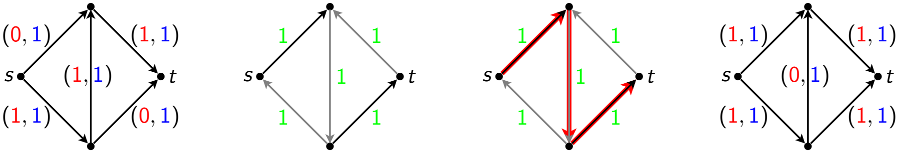


This flow is not of maximum value and no (s, t)-path exists in G'.

Flow augmentation

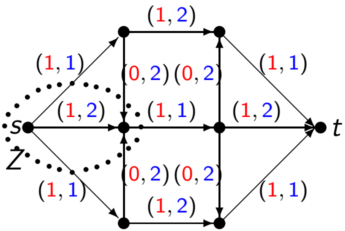
Second idea

1. Use the arcs uv such that $x(uv) > 0$ in the reverse direction.
2. $G_x := (V, A_x^1 \cup A_x^2)$ where $A_x^2 := \{vu : uv \in A, x(uv) > 0\}$.
3. If there exists an (s,t)-path P in G_x then we can augment the value of the flow by $\varepsilon_x := \min\{\varepsilon_x^1, \varepsilon_x^2\}$, where
   $\varepsilon_x^2 := \min\{x(uv) : vu \in A(P) \cap A_x^2\}$,
4. $x'(uv) := \begin{cases} x(uv) + \varepsilon_x & \text{if } uv \in A(P) \cap A_x^1 \\ x(uv) - \varepsilon_x & \text{if } vu \in A(P) \cap A_x^2 \\ x(uv) & \text{otherwise.} \end{cases}$



Theorem (Ford-Fulkerson)

1. A feasible (s,t)-flow x is of maximum value if and only if there exists no (s,t)-path in G_x.
2. $\max\{val(x):$ feasible (s,t)-flow $x\} = \min\{cap(Z):$ (s,t)-cut $Z\}$.



Proof

Proof of necessity

1. Suppose there exists an (s,t)-path P in G_x.
2. $$x'(uv) := \begin{cases} x(uv) + \varepsilon_x & \text{if } uv \in A(P) \cap A_x^1 \\ x(uv) - \varepsilon_x & \text{if } vu \in A(P) \cap A_x^2 \\ x(uv) & \text{otherwise.} \end{cases}$$
3. x' is a feasible (s,t)-flow of value $\text{val}(x) + \varepsilon_x > \text{val}(x)$.
    1. $\varepsilon_x > 0$,
    2. x' is an (s,t)-flow,
    3. x' is feasible,
    4. $\text{val}(x') = \text{val}(x) + \varepsilon_x$.
4. x is not of maximum value, contradiction.

Proof

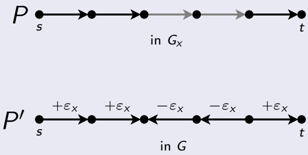
Proof of necessity

1. $\varepsilon_x > 0$:

   1. $g(uv) - x(uv) > 0 \ \forall uv \in A_x^1$, hence
      $\varepsilon_x^1 = \min\{g(uv) - x(uv) : uv \in A(P) \cap A_x^1\} > 0$,
   2. $x(uv) > 0 \ \forall vu \in A_x^2$, hence
      $\varepsilon_x^2 = \min\{x(uv) : vu \in A(P) \cap A_x^2\} > 0$,
   3. thus $\varepsilon_x = \min\{\varepsilon_x^1, \varepsilon_x^2\} > 0$.

Proof

Proof of necessity



2. x' is an (s,t)-flow:

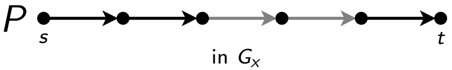
1. $x_{P'}(uv) := \begin{cases} +\varepsilon_x & \text{if } uv \in A(P) \cap A_x^1 \\ -\varepsilon_x & \text{if } vu \in A(P) \cap A_x^2 \\ 0 & \text{otherwise.} \end{cases}$ is an (s,t)-flow.
2. $x' = x + x_{P'}$.
3. $$\begin{aligned} d^-_{x'}(v) - d^+_{x'}(v) &= d^-_{x+x_{P'}}(v) - d^+_{x+x_{P'}}(v) \\ &= (d^-_x(v) - d^+_x(v)) + (d^-_{x_{P'}}(v) - d^+_{x_{P'}}(v)) \ \forall v \neq s,t \\ &= 0 + 0 = 0 \end{aligned}$$
4. x' is hence an (s,t)-flot.

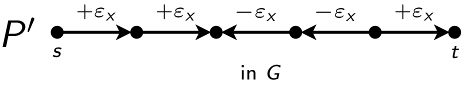
Proof

Proof of necessity

3. x' is feasible: Since x is feasible, $0 \le x(e) \le g(e)\ \forall e \in A$.

1. $e \in A \setminus A(P)$: $-x(e) \le 0 = x_{P'}(e) = 0 \le g(e) - x(e)$.
2. $e \in A_x^1 \cap A(P)$: $-x(e) \le 0 \le \varepsilon_x = x_{P'}(e) = \varepsilon_x \le \varepsilon_x^1 \le g(e) - x(e)$.
3. $\overleftarrow{e} \in A_x^2 \cap A(P)$: $-x(e) \le -\varepsilon_x^2 \le -\varepsilon_x = x_{P'}(e) = -\varepsilon_x \le 0 \le g(e) - x(e)$.
4. $e \in A$: $0 \le (x + x_{P'})(e) = x'(e) = (x + x_{P'})(e) \le g(e)$.
5. x' is hence feasible.





Proof

Proof of necessity

4. $\mathrm{val}(x') = \mathrm{val}(x) + \varepsilon_x$: Since $\delta^-_G(s) = \emptyset$, the first arc su of P belongs to A^1_x and hence to A.

$$
\begin{aligned}
val(x') = d^+_{x'}(s) &= \left(\sum_{sv \in \delta^+_G(s) \setminus su} x'(sv)\right) + x'(su) \\
&= \left(\sum_{sv \in \delta^+_G(s) \setminus su} x(sv)\right) + (x(su) + \varepsilon_x) \\
&= d^+_x(s) + \varepsilon_x \\
&= val(x) + \varepsilon_x.
\end{aligned}
$$

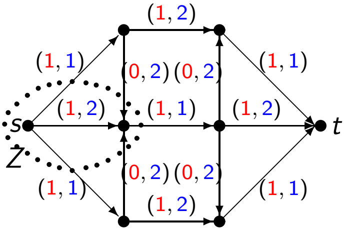
Proof

Proof of sufficiency

1. Suppose there exists no (s,t)-path in G_x.
2. $Z := \{v \in V : \exists$ an (s,v)-path in $G_x\} \Longrightarrow$
3. $s \in Z \subseteq V \setminus t$ and
4. $\delta^+_{G_x}(Z) = \emptyset. \Longrightarrow$
5. $\forall uv \in \delta^+_G(Z) : x(uv) = g(uv)$ and,
6. $\forall uv \in \delta^-_G(Z) : x(uv) = 0. \Longrightarrow$
7. $cap(Z) = d^+_g(Z) = d^+_x(Z) - d^-_x(Z) = val(x) \leq Max\ Flow \leq Min\ Cut \leq cap(Z), \Longrightarrow$
8. We have hence equality everywhere, in particular,
9. the flow x is of maximum value and
10. Max Flow = Min Cut.

Algorithm of Edmonds-Karp

INPUT : Network (G, g) such that $g \geq 0$, $s, t \in V$: $\delta^-(s) = \emptyset = \delta^+(t)$.
OUTPUT : feasible (s,t)-flow x and (s,t)-cut Z such that $val(x) = cap(Z)$.



Algorithm

Step 0: $x_0(e) = 0 \ \forall e \in A$, $i := 0$.

Step 1: Construct the auxiliary graph $G_i := (V, A_i^1 \cup A_i^2)$ where
$A_i^1 := \{uv : uv \in A, x_i(uv) < g(uv)\}$ and
$A_i^2 := \{vu : uv \in A, x_i(uv) > 0\}$.

Step 2: Execute algorithm Breadth First Search on G_i and s to get $Z_i \subseteq V$ and an s-arborescence F_i of $G_i[Z_i]$ such that $\delta^+_{G_i}(Z_i) = \emptyset$.

Step 3: If $t \notin Z_i$ then stop with $x := x_i$ and $Z := Z_i$.

Step 4: Otherwise, $P_i := F_i[s,t]$, the unique (s,t)-path in F_i.

Step 5: $\varepsilon_i^1 := \min\{g(uv) - x_i(uv) : uv \in A(P_i) \cap A_i^1\}$,
$\varepsilon_i^2 := \min\{x_i(uv) : vu \in A(P_i) \cap A_i^2\}$,
$\varepsilon_i := \min\{\varepsilon_i^1, \varepsilon_i^2\}$.

Step 6: $x_{i+1}(uv) := \begin{cases} x_i(uv) + \varepsilon_i & \text{if } uv \in A(P_i) \cap A_i^1 \\ x_i(uv) - \varepsilon_i & \text{if } vu \in A(P_i) \cap A_i^2 \\ x_i(uv) & \text{otherwise.} \end{cases}$

Step 7: $i := i + 1$ and go to Step 1.

Complexity of the algorithm

Theorem

The algorithm of Edmonds-Karp stops in polynomial time.

Remark

1. Since the algorithm BFS is executed in Step 2, the algorithm always augments the flow on a shortest (s, t)-path in G_x.
2. If the algorithm BFS is replaced in Step 2 by an arbitrary search algorithm, then it may happen that the algorithm does not stop.

Integer Flows

Theorem

If $g(e)$ is integer for every arc e of G, then there exists a feasible flow x of maximum value such that $x(e)$ is integer for every arc e of G.

Proof

1. By executing the algorithm of Edmonds-Karp, we see by induction on i that every $x_i(e)$ is integer:
2. For $i = 0, x_0(e) = 0$ is integer $\forall e \in A$.
3. Suppose that it is true for i.
4. $\varepsilon_i = \min\{\varepsilon_i^1, \varepsilon_i^2\}$ is integer:
   1. $\varepsilon_i^1 = \min\{g(uv) - x_i(uv) : uv \in A(P_i) \cap A_i^1\}$ is integer: every $g(e) - x_i(e)$ is integer.
   2. $\varepsilon_i^2 = \min\{x_i(uv) : vu \in A(P_i) \cap A_i^2\}$ is integer: every $x_i(e)$ is integer.
5. $x_{i+1}(e), =$ either $x_i(e)$ or $x_i(e) + \varepsilon_i$ or $x_i(e) - \varepsilon_i$, is integer $\forall e \in A$.

Applications

Citation

"Knowledge is useless without consistent application." - Julian Hall

Applications of integer flows

1. Menger's Theorem on connectivity,
2. Kőnig's Theorem on matchings.

Applications of flows and cuts

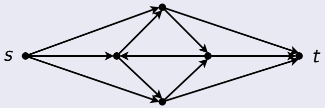
1. Open pit mining,
2. Distributed computing on a two-processor computer,
3. Image segmentation.

Exercises

Exercise 1

Let $G := (V, A)$ be a directed graph, $s, t \in V$ and $k \in \mathbb{Z}^+$ such that

$$\begin{aligned} d^+(s) - d^-(s) &= k, \\ d^+(t) - d^-(t) &= -k, \\ d^+(v) - d^-(v) &= 0 \quad \forall v \in V \setminus \{s, t\}. \end{aligned}$$



Prove that G admits k arc-disjoints directed (s, t)-paths.

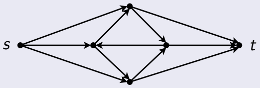
Exercise 2

Given a directed graph $G = (V, A)$, $s, t \in V$ and a non-negative integer (s, t)-flow x, prove that G contains $val(x)$ directed (s, t)-paths such that each arc a belongs to at most $x(a)$ of the paths.

Exercise 3

Theorem of Menger

Given a directed graph $G = (V, A)$ and $s, t \in V$,
maximum number of arc-disjoint (s, t)-paths $=$
minimum out-degree of an (s, t)-cut.



Ford-Fulkerson $\Longrightarrow$ Menger

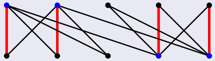
Let $G' := G - \delta^+(t) - \delta^-(s)$ and $g(e) := 1 \; e \in A(G')$.

(a) Prove that max $=$ maximum value of a feasible (s, t)-flow in (G', g).
(b) Prove that min $=$ minimum capacity of an (s, t)-cut in (G', g).
(c) Deduce Menger's Theorem from (a), (b) and the integer version of Ford-Fulkerson's Theorem.

Exercise 4

Theorem of Kőnig:

Given a bipartite graph $G = (U, V; E)$,
maximum cardinality of a matching of G =
minimum cardinality of a transversal of G.



Ford-Fulkerson $\Longrightarrow$ Kőnig

Let $(D := (W, A), g)$ be a network where $W := U \cup V \cup \{s, t\}$,
$A := \{su : u \in U\} \cup \{vt : v \in V\} \cup \{uv : u \in U, v \in V, uv \in E\}$,
$g(su) := 1 \ \forall u \in U$, $g(vt) := 1 \ \forall v \in V$ and $g(uv) := |U| + 1 \ \forall uv \in E$,

x an integer feasible (s, t)-flow of max. value, Z an (s, t)-cut of min. capacity,
$M := \{uv \in E : x(uv) = 1\}$ and $T := (U - Z) \cup (V \cap Z)$.

(a) Prove that M is a matching of G of size $val(x)$.

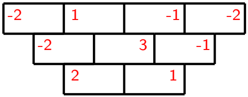
(b) Prove that T is a transversal of G of size $cap(Z)$.

(c) Deduce Kőnig Theorem from (a), (b) and Ford-Fulkerson Theorem.

Applications: Open pit mining

Open pit mining

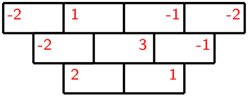
- A company wants to exploit an open pit mining
  - by removing blocks
  - to maximize the profit.
    - A block can be removed only if the blocks lying above it have already been removed.
- Each block has a net profit obtained from removing it.
  - This value can be positive or negative, it depends on the cost of
    - exploiting the block and
    - the richness of its contents.
- We show how to model the problem by a problem of minimum capacity cut in a network.

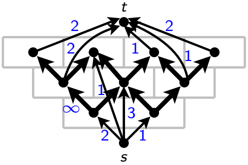


Open pit mining

Model

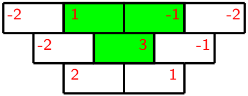
1. $p(v)$:= profit of the block v,
2. P := blocks of positive profit,
3. N := blocks of negative profit,
4. Network $(G := (V, A), g)$ where
   1. $V := P \cup N \cup \{s, t\}$,
   2. $A :=$ {the arcs of constraint} $\cup \{sv : v \in P\} \cup \{ut : u \in N\}$,
   3. $g(uv) := \begin{cases} \infty & \text{if } uv \text{ arc of constraint,} \\ p(v) & \text{if } u = s, \\ -p(u) & \text{if } v = t. \end{cases}$

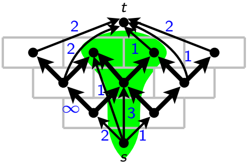




Lemma

The blocks in B satisfy the removal contraint $\iff$ $cap(B \cup s) < \infty$.
(By construction.)



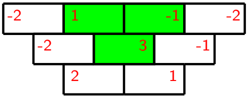


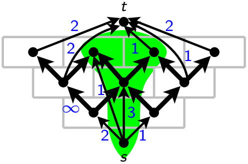
Open pit mining

Lemma

$$
\begin{aligned}
cap(B \cup s) &= \sum_{v \in P \setminus B} g(sv) + \sum_{v \in N \cap B} g(vt) = \sum_{v \in P \setminus B} p(v) + \sum_{v \in N \cap B} -p(v) \\
&= \Big(\sum_{v \in P} p(v) - \sum_{v \in P \cap B} p(v)\Big) - \Big(\sum_{v \in B} p(v) - \sum_{v \in P \cap B} p(v)\Big) \\
&= \sum_{v \in P} p(v) - \sum_{v \in B} p(v).
\end{aligned}
$$

minimize $=$ constant $-$ maximize





Distributed computing on a two-processor computer

- Assign the modules of a program to two processors in a way that
  - minimizes the total cost of
    - computation and
    - interprocessor communication.
- We know in advance
  - for each module, its computation cost on each of the two processors,
  - for each pair of modules, their interprocessor communication cost
    - if they are assigned to different processors.
- We show how to model the problem by a problem of minimum capacity cut in a network.

Distributed computing on a two-processor computer

computation cost

	M_1	M_2	M_3	M_4	
P_1	4	4	1	2	a_i
P_2	1	2	4	4	b_i

communication cost

	M_1	M_2	M_3	M_4	
M_1	0	5	1	1	
M_2	5	0	1	1	c_{ij}
M_3	1	1	0	5	
M_3	1	1	5	0	

Cost to minimize

total cost $=$

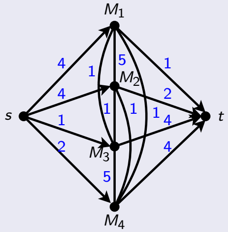
computation cost of modules executed on P_1 (C_1) $+$
computation cost of modules executed on P_2 (C_2) $+$
communication cost for the pair of modules executed on differents processors

$$= \sum_{M_i \in C_1} a_i + \sum_{M_j \in C_2} b_j + \sum_{M_i \in C_1, M_j \in C_2} c_{ij}.$$

Model

Network $(G := (V, A), g)$ where

1. $V := \{M_1, M_2, M_3, M_4, s = P_1, t = P_2\}$,
2. $A := \{uv, vu : u, v \in V \setminus \{s, t\}\}$
   $\cup \{sv : v \in V \setminus \{s, t\}\}$
   $\cup \{vt : v \in V \setminus \{s, t\}\}$,
3. $g(uv) := \begin{cases} c_{ij} & \text{if } uv = M_iM_j, \\ a_i & \text{if } uv = sM_i, \\ b_j & \text{if } uv = M_jt. \end{cases}$



	M_1	M_2	M_3	M_4	
P_1	4	4	1	2	a_i
P_2	1	2	4	4	b_i

	M_1	M_2	M_3	M_4	
M_1	0	5	1	1	
M_2	5	0	1	1	c_{ij}
M_3	1	1	0	5	
M_3	1	1	5	0	

Distributed computing on a two-processor computer

Lemme

$$
\begin{aligned}
cap(C_2 \cup s) &= \sum_{M_i \in C_1} g(sM_i) + \sum_{M_j \in C_2} g(M_j t) + \sum_{M_i \in C_1, M_j \in C_2} g(M_i M_j) \\
&= \sum_{M_i \in C_1} a_i + \sum_{M_j \in C_2} b_j + \sum_{M_i \in C_1, M_j \in C_2} c_{ij},
\end{aligned}
$$

which is the total cost to minimize (C_i = the modules executed on P_i).

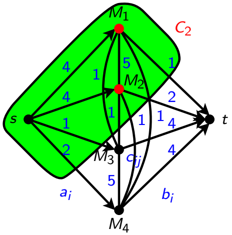
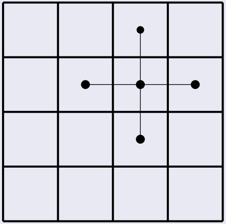


Image segmentation

- We have to locate objects in a digital image.
- Every $i \in V$, where V is the set of pixels of the image,
  - belongs to an object with likelihood p_i and
  - belongs to the background with likelihood q_i.
- We also have a penalty function $r(i,j)$ of separation for every pair $(i,j) \in E$ where E is the set of pairs of neighboring pixels.
- We have to find a partition S, T of V that maximizes

$$\sum_{i \in S} p_i + \sum_{j \in T} q_j - \sum_{i \in S, j \in T, (i,j) \in E} r(i,j).$$



- We show how to model the problem by a problem of minimum capacity cut in a network.

Image segmentation

Model

Network $(G = (V', A), g)$ where

1. $V' := V \cup \{s, t\}$,
2. $A := \{uv, vu : uv \in E\} \cup \{sv : v \in V\} \cup \{vt : v \in V\}$,
3. $g(uv) := \begin{cases} p_i & \text{if } uv = si, \\ q_j & \text{if } uv = jt, \\ r(i,j) & \text{if } uv = ij \in E. \end{cases}$

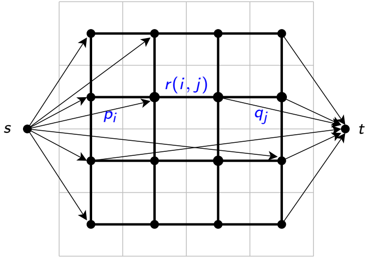


Image segmentation

Lemma

$$
\begin{aligned}
cap(S \cup s) &= \sum_{j\in T} g(sj) + \sum_{i\in S} g(it) + \sum_{i\in S, j\in T, ij\in E} g(ij) \\
&= \sum_{j\in T} p_j + \sum_{i\in S} q_i + \sum_{i\in S, j\in T, ij\in E} r(i,j) \\
&= \sum_{i\in V}(p_i + q_i) - (\sum_{i\in S} p_i + \sum_{j\in T} q_j - \sum_{i\in S, j\in T, ij\in E} r(i,j)).
\end{aligned}
$$

minimize = constant − maximize