Discrete Optimization

Quentin Louveaux

ULg - Institut Montefiore

2016

The maximum flow problem

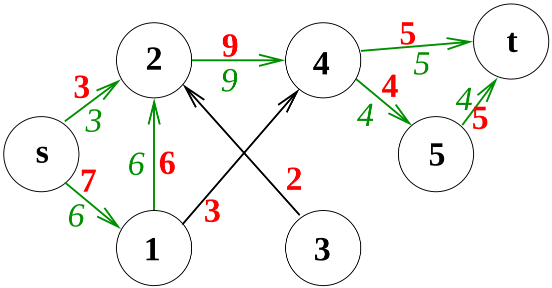
Let $G = (V, E)$ be a directed graph with a source node s and a sink node t and with capacities u_j on each edge e_j.

The maximum flow problem

Find the maximum flow sent from s to t where the flow entering each node (except s and t) is equal to the flow leaving each node and where the flow on each edge satisfies the capacity.

The max flow problem

Send the maximum flow from a node *s* (source) to a node *t* (sink).
Each edge has a capacity
Example of a feasible flow



The minimum cut problem

The minimum $s - t$-cut problem

An $s - t$-cut is a partition of the nodes of G into V_1 and V_2 where $s \in V_1$ and $t \in V_2$.
The value of an $s - t$-cut is equal to the sum of the capacities of all edges joining one node of V_1 to one node of V_2.
The min-cut problem is the problem of finding the $s - t$-cut that has the minimal value.

Important result

Max flow = min cut

Examples of applications

The problem of representatives

A town has r residents $R_1, \ldots, R_r$, q clubs $C_1, \ldots, C_q$ and p political parties $P_1, \ldots, P_p$. Each resident is member of at least one club and can belong to exactly one political party.

Each club must nominate one of its members to represent it in the town's council in such a way that the number of council members belonging to the political party P_k is at most u_k.

Problem with applications in resource assignement settings.

Examples of applications

Matrix rounding problem

We are given a matrix $D \in \mathbb{R}^{p \times q}$ with row sums α_i and column sums β_j. We can decide to round any element in the matrix to the lower or the upper integer.

How can we round each element such that, for each row and each column, the sum of the rounded elements is equal to a rounding of the sums.

This is called a consistent rounding.

Examples of applications

Scheduling on uniform parallel machines

We want to schedule J jobs on M parallel machines.
Each job j has a processing time p_j, a release date r_j and a due date d_j. Furthermore we allow preemption.

Examples of applications for the min-cut problem

Distributed computing on a 2-processor computer

Consider a large computer program that has to be run on two processors. The code is subdivided in tasks that have a different processing time on both processors. Furthermore there is a communication cost if some tasks are performed on different processors.

Example of application of the min-cut problem

Segmentation of an image

Consider a picture taken of of an object : for example, an MRI image of a heart. We may want to detect, for each pixel, whether it actually belongs to the heart or whether it belongs to the background.

It is possible to create a similarity function for each pair of pixel that is large if the 2 pixels are likely to belong to the same part of the picture.

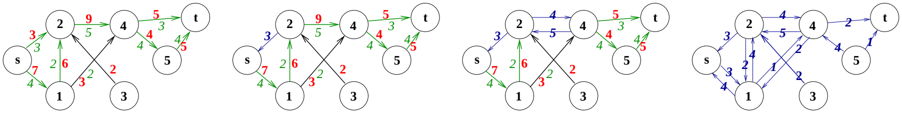
Flows and cuts

The residual network

Given a flow x, the residual capacity r_{ij} is the maximum additional flow that can be added to arc (i,j) or removed from arc (j,i).

$$r_{ij} = (u_{ij} - x_{ij}) + x_{ji}$$

This yields the graph $G(x)$, the residual network.



$(s - t)-$cuts

Definition of an $(s - t)-$cut

A cut is a partition of the node set into two subsets $S, \bar{S}$ such that

- $S \cup \bar{S} = V$
- $S \cap \bar{S} = \emptyset$
- It is an $(s - t)-$cut if $s \in S$ and $t \in \bar{S}$.

The capacity of an $(s - t)-$cut is the sum of the capacities of the forward arcs of the cut :

$$u(S, \bar{S}) = \sum_{(i,j) \in (S, \bar{S})} u_{ij}.$$

The residual capacity of an $(s - t)-$cut is the sum of the residual capacities of the forward arcs of the cut :

$$u(S, \bar{S}) = \sum_{(i,j) \in (S, \bar{S})} r_{ij}.$$

Property : The value of any flow is less or equal to the capacity of any cut in the network. This is a weak duality property.

Generic augmenting path algorithm

Definition

A directed path from the source s to the sink t in the residual network is an augmenting path.
The residual capacity of the augmenting path is the minimum residual capacity over all arcs in the path.

Idea : augment step by step the value of the current flow using paths in the residual network.

The value by which we can augment the flow is less or equal to the residual capacity of any cut.

Generic augmenting path algorithm

```
x:=0;
while $G(x)$ contains a directed path from $s$ to $t$ do
    Find an augmenting path $P$
    $\delta := \min\{r_{ij} \mid (i,j) \in P\}$
    Augment $\delta$ units of flow along $P$ and update $G(x)$
```

Important questions :
Does the algorithm terminate ?
Does it provide an optimal solution ?
What is its running time ?

Labeling algorithm

This is a first implementation of the augmenting path algorithm that does not run in polynomial time.

Idea : Perform a breadth-first-search or a depth-first-search on the graph to label the nodes. If the sink node is labeled, we have found an augmenting path.

The algorithm is correct

If, at an iteration, the sink node cannot be labeled, then the flow is maximal.

The max flow-min cut theorem

Theorem

The maximum value of of the flow from s to t equals the minimum capacity among all $(s - t)-$cuts.

Augmenting path theorem

A flow x^* is a maximum flow if and only if the residual network $G(x^*)$ contains no augmenting path.

Integrality theorem

If all arc capacities are integer, the maximum flow problem has an integer maximum flow.

Complexity of the labeling algorithm

Theorem

The labeling algorithm solves the maximum flow problem in $\mathcal{O}(mnU)$ operations.

This implies that in a such an implementation, the algorithm does not run in polynomial time in the worst case.

Combinatorial implications

- The maximum number of arc-disjoint paths from s to t is equal to the minimum number of arcs whose removal disconnects s from t.
- The maximum number of node-disjoint paths from s to t is equal to the minimum number of nodes whose removal disconnects s from t.
- Matchings and covers are strong dual for bipartite graphs.

Flows with lower bounds

Optimizing from a feasible flow

Just change the computation of the residual capacity

$$r_{ij} = (u_{ij} - x_{ij}) + (x_{ji} - l_{ji})$$

Establishing a feasible flow

- Add an arc (t, s) of infinite capacity and transform the problem into finding a feasible circulation.
- At each node, compute
$$b(i) = \sum_{j|(j,i)\in A} l_{ji} - \sum_{j|(i,j)\in A} l_{ij}$$
- Put only upper bounds on the arcs to find x'
- Create a dummy source node s' and a dummy sink node t'.
- Create an edge from s' to i if $b(i) > 0$ with capacity $b(i)$.
- Create an edge from i to t' if $b(i) < 0$ with capacity $|b(i)|$.
- The problem is feasible if there exists a flow with total value $\sum_i b(i)$
- The potential feasible flow is obtained as $x_{ij} := x'_{ij} + l_{ij}$