Graph Algorithms - Lecture 8

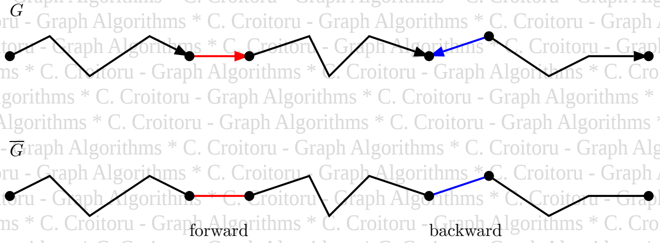
November 29, 2024

Table of contents

Maximum flow problem - Augmenting paths

Definition

Let P be a path in $\widetilde{G}$ – the support graph of the digraph G *– and $e = v_iv_j$ an edge of P, $e \in E(P)$. If e corresponds to the arc v_iv_j then e is a* forward arc *of P, otherwise (e corresponds to the arc v_jv_i) e is a* backward arc *of P.*



Definition

Let $R = (G, s, t, c)$ be a flow network and x be a flow in R. An A-path *(in R with respect to the flow x) is a path P in $\widetilde{G}$ such that $\forall ij \in E(P)$:*

- *if ij is a forward arc, then $x_{ij} < c_{ij}$,*
- *if ij is a backward arc, then $x_{ji} > 0$.*

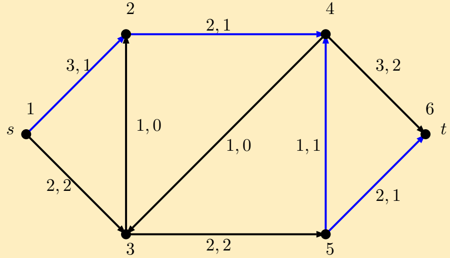
If P is an A-path in R w. r. t. flow x, then the residual capacity *of $ij \in E(P)$ is*

$$r(ij) = \begin{cases} c_{ij} - x_{ij}, & \text{if } ij \text{ is a forward arc of } P \\ x_{ji}, & \text{if } ij \text{ is a backward arc of } P \end{cases}$$

The residual capacity *of the path P is $r(P) = \min\limits_{e \in E(P)} r(e)$.*

Example

Let us consider the flow network below, where, on each edge, the first label is its capacity and the second label is the corresponding flow.



P : $1, 12, 2, 24, 4, 45, 5, 56, 6$ is an A-path from s to t with forward arcs 12 ($x_{12} = 1 < c_{12} = 3$), 24 ($x_{24} = 1 < c_{24} = 2$), 56 ($x_{56} = 1 < c_{56} = 2$), and backward arc 45 ($x_{45} = 1 > 0$). Residual capacity of P: $r(P) = \min\{2, 1, 1, 1\} = 1$.

Definition

An augmenting path *w.r.t the flow x in the flow network $R = (G, s, t, c)$ is an A-path from s to t.*

Lemma 1

If P is an augmenting path of the flow x in $R = (G, s, t, c)$, then $x^1 = x \otimes r(P)$ defined by

$$x^1_{ij} = \begin{cases} x_{ij}, & \text{if } ij \notin E(P) \\ x_{ij} + r(P), & \text{if } ij \in E(P), ij \text{ forward arc in } P \\ x_{ij} - r(P), & \text{if } ij \in E(P), ij \text{ backward arc in } P \end{cases}$$

is a flow in R with $v(x^1) = v(x) + r(P)$.

Proof. By the definition of $r(P)$, the constraints (i) are fulfilled by x^1. The constraints (ii) verified by x - are not affected for x^1 in a vertex $i \notin V(P)$. For each $i \in V(P) \setminus \{s,t\}$ there are exactly two arcs of P incident with i, say li and ik. We have the following possible cases:

a) li and ik are forward arcs:

$$\sum_j x^1_{ji} - \sum_j x^1_{ij} = \sum_{j \neq l} x_{ji} - \sum_{j \neq k} x_{ij} + x^1_{li} - x^1_{ik}$$

$$= \sum_{j \neq l} x_{ji} - \sum_{j \neq k} x_{ij} + x_{li} + r(P) - x_{ik} - r(P) = \sum_j x_{ji} - \sum_j x_{ij} = 0.$$

b) li is a forward arc and ik is a backward arc:

$$\sum_j x^1_{ji} - \sum_j x^1_{ij} = \sum_{j \neq l,k} x_{ji} - \sum_j x_{ij} + x^1_{li} + x^1_{ki}$$

$$= \sum_{j \neq l,k} x_{ji} - \sum_j x_{ij} + x_{li} + r(P) + x_{ki} - r(P) = \sum_j x_{ji} - \sum_j x_{ij} = 0.$$

c) li is a backward arc and ik is a forward arc: likewise to b).

d) li and ik are both backward arcs: likewise to a).

$v(x^1)$ differs from $v(x)$ because of the flow on the arc lt of P:

lt is a forward arc:

$$v(x^1) = \sum_j x^1_{jt} - \sum_j x^1_{tj} = \sum_{j \neq l} x_{jt} - \sum_j x_{tj} + x^1_{lt} =$$

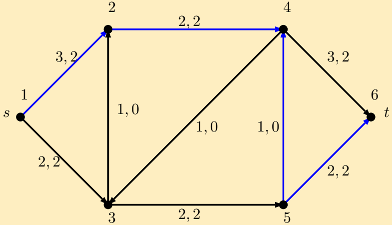
$$= \sum_{j \neq l} x_{jt} - \sum_j x_{tj} + x_{lt} + r(P) = v(x) + r(P).$$

lt is a backward arc:

$$v(x^1) = \sum_j x^1_{jt} - \sum_j x^1_{tj} = \sum_j x_{jt} - \sum_{j \neq l} x_{tj} - x^1_{tl} =$$

$$= \sum_j x_{jt} - \sum_{j \neq l} x_{tj} - (x_{lt} - r(P)) = v(x) + r(P). \square$$

For the above example, the flow $x^1 = x \otimes r(P)$ which has the value $v(x^1) = v(x) + r(P) = 3 + 1 = 4$ is:



Remarks

- The above lemma explains the names for augmenting paths and residual capacity.
- By the definition, if P is an augmenting path, then $r(P) > 0$ and thus $v(x \otimes r(P)) > v(x)$. It follows that if there is an augmenting path of the flow x, then x is not a flow of maximum value.

Definition

Let $R = (G, s, t, c)$ be a flow network. A cut *in R is any partition (S, T) of $V(G)$ with $s \in S$ and $t \in T$. The* capacity of the cut (S, T) *is*

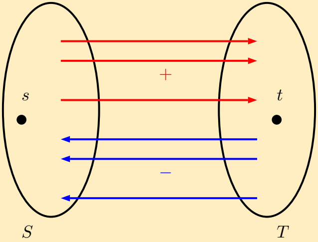
$$c(S, T) = \sum_{i \in S} \sum_{j \in T} c_{ij}.$$

Lemma 2

If x is a flow in $R = (G, s, t, c)$ and (S, T) is a cut in this network, then

$$v(x) = \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji})$$

(the value of the flow is the net flow passing through this cut).



Proof.

$$v(x) = \left(\sum_j x_{sj} - \sum_j x_{js}\right) + 0 =$$

$$= \left(\sum_j x_{sj} - \sum_j x_{js}\right) + \sum_{i \in S, i \neq s} \left(\sum_j x_{ij} - \sum_j x_{ji}\right) =$$

Proof cont'd.

$$= \sum_{i \in S} \left(\sum_j x_{ij} - \sum_j x_{ji} \right) = \sum_{i \in S} \sum_j (x_{ij} - x_{ji}) =$$

$$= \sum_{i \in S} \sum_{j \in S} (x_{ij} - x_{ji}) + \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}) =$$

$$= 0 + \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}). \ \square$$

Lemma 3

If x is a flow in $R = (G, s, t, c)$ and (S, T) is a cut in this network, then

$$v(x) \leqslant c(S, T).$$

Proof. By Lemma 2

$$v(x) = \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}) \overset{x_{ij} \leqslant c_{ij}}{\leqslant} \sum_{i \in S} \sum_{j \in T} (c_{ij} - x_{ji}) \overset{x_{ji} \geqslant 0}{\leqslant}$$

$$\overset{x_{ji} \geqslant 0}{\leqslant} \sum_{i \in S} \sum_{j \in T} c_{ij} = c(S, T). \square$$

Remark

If $\overline{x}$ is a flow in R and $(\overline{S}, \overline{T})$ is a cut such that $v(\overline{x}) = c(\overline{S}, \overline{T})$, then, $\forall x$ flow in R, we have $v(x) \leqslant c(\overline{S}, \overline{T}) = v(\overline{x})$, i. e., $\overline{x}$ is a maximum value flow in R.

Similarly, $\forall (S, T)$ cut in R, we have $c(S, T) \geqslant v(\overline{x}) = c(\overline{S}, \overline{T})$, that is, $(\overline{S}, \overline{T})$ is a minimum capacity cut in R.

Theorem 1

A flow x is a flow of maximum value if and only if there is no augmenting path w.r.t. x in R.

Proof. "$\Rightarrow$" If P is an augmenting path w. r. t. x, then $x \otimes r(P)$ is a flow in R of strictly greater value.
"$\Leftarrow$" Let x be a flow in R with the property that there is no augmenting path w.r.t. x in R. Let

$$S = \{i \,:\, i \in V \text{ and } \exists P \text{ an } A\text{-path in } R \text{ from } s \text{ to } i\}.$$

Obviously $s \in S$ (there exists an A-path of length 0 from s to s) and $t \notin S$ (there is no A-path from s to t). Hence, if $T = V \setminus S$, then (S, T) is a cut in R.

Proof cont'd. $\forall i \in S$ and $\forall j \in T$ we have:

- if $ij \in E$, then $x_{ij} = c_{ij}$ and
- if $ji \in E$, then $x_{ji} = 0$

(otherwise the A-path from s to i can be extended to an A-path from s to j, hence $j \in S$ - contradiction).

It follows that $v(x) = \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}) = \sum_{i \in S} \sum_{j \in T} c_{ij} = c(S, T)$, i. e., x is a flow of maximum value (see the above remark). $\square$

Theorem 2

(Integral flow theorem) If all capacities in R are integer then there exists an integral flow, x, of maximum value (all $x_{ij} \in \mathbb{Z}_+$).

Proof. Let us consider the following algorithm:

$x^0 \leftarrow 0$; $i \leftarrow 0$;
while ($\exists P_i$ an augmenting path w. r. t. x^i) do
  $x^{i+1} \leftarrow x^i \otimes r(P_i)$; $i++$;
end while

Note that "x^i has only integral components" is an invariant of the algorithm (from the definition of $r(P_i)$, if all capacities are integral and x^i is integral, then $r(P_i)$ is integral and hence x^{i+1} is integral).

Proof cont'd. Moreover, in each **while** iteration the value of the current flow increases (with at least one), therefore the algorithm terminates in at most $c(\{s\}, V \setminus \{s\}) \in \mathbb{Z}_+$ **while** iterations.
There is no augmenting paths w.r.t. the final flow, hence, by Theorem 1, is of maximum value. $\square$

Remark

The above algorithm terminates too, when the capacities are rational numbers.

Theorem 3

(Max flow - Min cut theorem) The maximum value of a flow in the flow network $R = (G, s, t, c)$ is equal to the minimum capacity of a cut in R.

Line of proof. If we devise an algorithm that, starting from an initial flow x^0 (e.g., $x^0 = 0$), constructs in finite time a flow x with respect to which there is no augmenting paths, then the cut considered in the proof of Theorem 1 satisfies together with x the required equality. For rational capacities, the algorithm considered in the proof of Theorem 2 satisfies this condition. For arbitrary real capacities we will present later such an algorithm due to Edmonds and Karp (1972).

Remarks

- A short proof of the above theorem is to show that there is a flow of maximum value and to apply the construction in the proof of Theorem 1. A flow of maximum value exists always by observing that it is the optimal solution of a linear program (over a non-empty polytope).
- The algorithmic importance of Theorem 3 is given by the fact that the set of all cuts in the network is finite, whereas the set of all flows is not finite.

The algorithm maintains a labeling of the network's vertices in order to find the augmenting paths with respect to the current flow x. When there is no augmenting path, the flow x is of maximum value.
Let $R = (G = (V, E), s, t, c)$ a network flow and x a flow in R.

The label of a vertex j, that has three components (e_1, e_2, e_3), has the following meaning: there exists an A-path from s to j, P, where $e_1 = i$ is the predecessor of j on this path, $e_2 \in \{forward, backward\}$ is the direction of arc ij, and $e_3 = r(P)$.

Initially s is labeled $(\cdot, \cdot, \infty)$. The other vertices receive labels by scanning the already labeled vertices:

procedure scan(i)
for ($j \in V$, unlabeled) do
  if ($ij \in E$ and $x_{ij} < c_{ij}$) then
    label j by $e = (i, forward, \min\{e_3[i], c_{ij} - x_{ij}\})$;
  end if
  if ($ji \in E$ and $x_{ji} > 0$) then
    label j by $e = (i, backward, \min\{e_3[i], x_{ji}\})$;
  end if
end for

The meaning of the components of labels is maintained by the procedure scan.

When, in the procedure scan, the vertex t is labeled, an augmenting path P, w.r.t. the current flow x, is discovered in the following way:

$r(P)$ is the component e_3 of the label of t, its vertices can be found in $\mathcal{O}(n)$ time by exploring the first component of the labels, and the change $x \otimes r(P)$ is done in this exploration by using the second component of the labels.

For the new flow, a new labeling based search is considered.

If all labeled vertices have been scanned and the vertex t has not been labeled, it follows that w. r. t. the current flow there is no augmenting path in R, therefore the current flow it is of maximum value. If S is the set of labeled vertices and $T = V \setminus S$, then (S, T) is a cut of minimum capacity in R.

start with an initial flow $x = (x_{ij})$ (e. g., $x = 0$)
$e(s) \leftarrow (\cdot, \cdot, \infty)$;
while ($\exists$ labeled unscanned vertices) do
  choose i a labeled unscanned vertex;
  scan(i);
  if (t has been labeled) then
    change the flow on the path given by the labels;
    erase all labels; $e(s) \leftarrow (\cdot, \cdot, \infty)$;
  end if
end while
$S \leftarrow \{i : i \in V, i \text{ is labeled}\}$;
$T \leftarrow V \setminus S$;

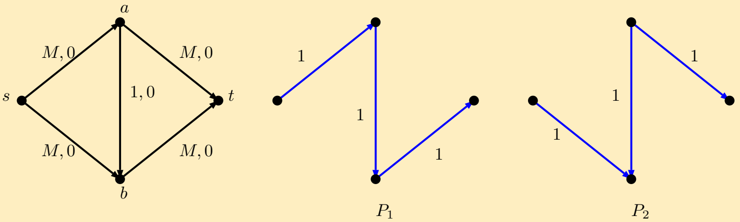
x is a maximum value flow, (S, T) is a minimum capacity cut.

Time complexity: Each flow augmenting requires at most $2m (m = |E|)$ arc inspections for doing vertex labeling. If all capacities are integers, at most v augmentations (v being the value of the maximum flow) are required. Hence the algorithm has $\mathcal{O}(mv)$ time complexity.
If U is an upper bound of all arc capacities, then $v \leqslant (n-1)U$ (this is an upper bound of the capacity of the cut $(\{s\}, V \setminus \{s\})$), therefore the algorithm has $\mathcal{O}(nmU)$ time complexity.

Remark

The algorithm may not terminate for irrational capacities. This case does not arise in practical implementations, but the weakness of the algorithm is given by the fact that the number of flow augmentings may depend on the capacities values (and not the size of the network).

Example



If the choose operation in the above algorithm makes that the succesive augmenting paths are $P_1, P_2, P_1, P_2, \ldots$, where $P_1 = (s, sa, a, ab, b, bt, t)$ and $P_2 = (s, sb, b, ba, a, at, t)$, then each residual capacity is 1, and the algorithm requires $2M$ flow augmentations, which is quite bad.

These drawbacks of the algorithm can be avoided if the choosing of the labeled vertices for scanning is made in a smart way (Dinic (1970), Edmonds & Karp (1972)).

Definition

A shortest augmenting path *w.r.t. the flow x in R is an augmenting path of minimum length over all augmenting paths w.r.t. x.*

Let x be a flow in the flow network R. Let x^k $(k \geqslant 1)$ be the flows from the sequence:

$x^1 \leftarrow x$;

$x^{k+1} \leftarrow x^k \otimes r(P_k)$, P_k being a shortest augmenting path w.r.t. x^k;

Edmonds & Karp modification of Ford & Fulkerson algorithm

In order to prove that this sequence is finite, let for each $i \in V$ and for each $k \in \mathbb{N}^*$:

$$\sigma_i^k = \text{ the minimum length of an } A\text{-path from } s \text{ to } i \text{ w. r. t. } x^k.$$

$$\tau_i^k = \text{ the minimum length of an } A\text{-path from } i \text{ to } t \text{ w. r. t. } x^k.$$

Lemma 4

$\forall i \in V$ and $\forall k \in \mathbb{N}^*$ we have

$$\sigma_i^{k+1} \geqslant \sigma_i^k \text{ and } \tau_i^{k+1} \geqslant \tau_i^k.$$

Proof. Omitted.

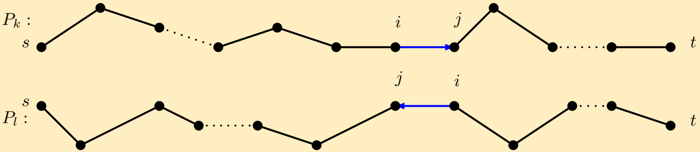
Theorem 4

(Edmonds & Karp, 1972) If $x = x^1$ is an arbitrary flow in the flow network R, then the sequence of flows $x^1, x^2, \ldots$, obtained from x^1 by successive shortest augmenting paths, has at most $mn/2$ terms (in at most $mn/2$ successive augmentations we obtain a flow x with the property that there is no augmenting path w.r.t. x).

Proof. If P is an augmenting path w.r.t. a flow x in R, with residual capacity $r(P)$, a critical arc in P is any arc $e \in E(P)$ with $r(e) = r(P)$. In $x \otimes r(P)$, the flow on critical arcs becomes either equal with capacity (on forward arcs), or null (on backward arcs).
Let ij be a critical arc on the shortest augmenting path P_k w.r.t. x^k. The length of P_k is $\sigma_i^k + \tau_i^k = \sigma_j^k + \tau_j^k$

Proof (cont'd). Since ij is a critical arc in P_k , in x^{k+1} it can not be used in the same direction as in P_k. Let P_l (with $l > k$) be the first shortest augmenting path w.r.t. x^l on which the flow on the arc ij will be modified, (when, the arc is used in the converse direction). We have two cases:

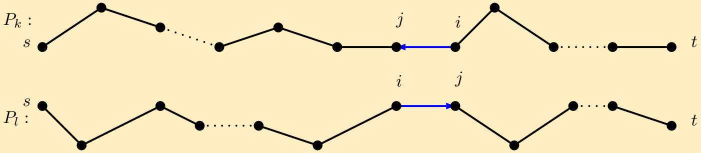
ij is a forward arc in P_k. Then $\sigma_j^k = \sigma_i^k + 1$, in P_l ij will be backward arc, therefore $\sigma_i^l = \sigma_j^l + 1$.



It follows that $\sigma_i^l + \tau_i^l = \sigma_j^l + 1 + \tau_i^l \geqslant \sigma_j^k + 1 + \tau_i^k = \sigma_i^k + \tau_i^k + 2$ (by Lemma 4). We obtained that $length(P_l) \geqslant length(P_k) + 2$.

Proof (cont'd).

ij is a backward arc in P_k. Then $\sigma_i^k = \sigma_j^k + 1$, in P_l ij will be forward arc, therefore $\sigma_j^l = \sigma_i^l + 1$.



It follows $\sigma_j^l + \tau_j^l = \sigma_i^l + 1 + \tau_j^l \geqslant \sigma_i^k + 1 + \tau_j^k = \sigma_j^k + \tau_j^k + 2$. We obtained that $length(P_l) \geqslant length(P_k) + 2$.

Hence any shortest augmenting path on which the arc ij is critical has the length with at least 2 greater than the length of the precedent path on which ij has been critical. Since the length of a path in G is at most $n-1$, it follows that a fixed arc can not be critical more than $n/2$ times (in the whole augmentation process).

Any augmenting path has at least one critical arc. Hence in the sequence (P_k) we have at most $mn/2$ shortest augmenting paths.

Corollary

In any flow network there is a flow x with the property that there is no augmenting path w.r.t. x.

Remarks

- The proof of the Theorem 4 is now completed.
- The only change in the Ford & Fulkerson algorithm is the choice of the labeled vertex which will be scanned: use the rule "first labeled-first scanned" that is, maintain a queue of labeled vertices initialized at each flow augmentation with the source s.

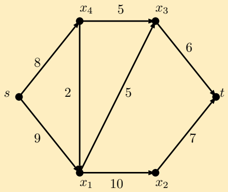
Summarizing, we have the following theorem.

Theorem 5

(Edmonds-Karp, 1972) If in the Ford & Fulkerson algorithm the scan of the labeled vertices is made in a bfs manner, then a flow of maximum value is obtained in $\mathcal{O}(m^2 n)$ time.

Proof. There are $\mathcal{O}(mn)$ flow augmentations (by Theorem 4), each requiring $\mathcal{O}(m)$ time. $\square$

Exercise 1. Consider the following network flow:



Find a maximum flow by performing the Edmonds-Karp algorithm on the following orderings of its adjacency lists:

(a) $A(s) = (x_1, x_4)$, $A(x_1) = (x_3, x_2, x_4)$, $A(x_2) = (x_1, t)$, $A(x_3) = (x_1, x_4, t)$, $A(x_4) = (x_3, x_1)$.

(b) $A(s) = (x_4, x_1)$, $A(x_1) = (x_2, x_3, x_4)$, $A(x_2) = (x_1, t)$, $A(x_3) = (x_1, x_4, t)$, $A(x_4) = (x_3, x_1)$.

,

Exercise 2. Consider a network $R = (G, s, t, ; c)$, where $G = (V, E)$ has n vertices and m arcs, and the capacity function has only integer values ($c : E \to \mathbb{Z}_+$). Let $C = \max_{e \in E} c(e)$.

(a) Prove that the maximum value of a flow in R is at most $m \cdot C$.

(b) Show that, for every flow x of R, and for every $K \in \mathbb{Z}_+$, we can find an augmenting path P with residual capacity, $r(P)$, at least K - if such a path exists - in $\mathcal{O}(m)$ time complexity.

Exercise 2 (cont'd).

(c) We consider the following algorithm:

SC-MAX-FLOW(R) {

  $C \leftarrow \max_{e \in E} c(e)$;

  $x \leftarrow 0$; // x is the current flow;

  $K \leftarrow 2^{1+\lfloor \log C \rfloor}$;

  while($K \geqslant 1$) {

    while(x has an augmenting path P with $r(P) \geqslant K$)

      $x \leftarrow x \otimes r(P)$;

    $K \leftarrow K/2$;

  }

  return x;

}

Exercise 2 (cont'd).

1. Prove that procedure SC-MAX-FLOW(R) returns a maximum value flow x in R.
2. Prove that, after every exterior **while** iteration, the maximum value of a flow in R is at most $v(x) + m \cdot K$.
3. Show that, $\forall\, K \in \mathbb{Z}_+$, there are at most $\mathcal{O}(m)$ interior **while** iterations. As a consequence the entire procedure has $\mathcal{O}(m^2 \log C)$ time complexity.

Exercise 3. The digraph $G = (V, E)$ represents the topology of a network of processors. For every processor, $v \in V$, we know its work load, $load(v) \in \mathbb{R}_+$. With the aid of a maximum flow in a certain network find a static load balancing strategy in G: indicate for any processor the work load which has to be send and to what processors in such a way that all the processors have the same work load.

Exercise 4. Let $R = (G, s, t, c)$ a network and (S_i, T_i), $i = \overline{1,2}$ two minimum capacity cuts in R. Show that $(S_1 \cup S_2, T_1 \cap T_2)$ and $(S_1 \cap S_2, T_1 \cup T_2)$ are minimum capacity cuts also.

Exercise 5. Let $R = (G = (V, E), s, t, c)$ be a network and $c : E \to \mathbb{Z}_+$, $n = |V|$, $m = |E|$.

(a) Devise an algorithm of $\mathcal{O}(n + m)$ time complexity for finding a minimum capacity cut in R, having at hand a maximum value flow x^*.

(b) Using a maximum flow algorithm for a suitable capacity, prove that you can find a minimum capacity cut in R, (S_0, T_0), having a minimum number of arcs.

Exercise 6. Let $G = (S, T; E)$ be a bipartite graph. Prove Hall's theorem (*it exists a matching in G which saturates the nodes from S if and only if (H)* $\forall A \subseteq S, |N_G(A)| \geqslant |A|$) by using the *maximum flow - minimum cut* theorem on a particular network.

Exercise 7. Let $R = (G, s, t, c)$ be a network having all arc capacities positive integer even numbers. Prove that it must exist a maximum flow having all values integer even numbers.

Exercise 8. Let $R = (G, s, t, c)$ be a network such that $st, ts \notin E(G)$. Furthermore we have a lower bound function $m : E(G) \to \mathbb{R}_+$, $m(e) \leqslant c(e), \forall\, e \in E(G)$. A flow φ in R is called *legal flow in R* if $x(e) \geqslant m(e), \forall\, e \in E(G)$.

(a) Prove that, for every legal flow φ and every st-cut (S, T) we have

$$v(\varphi) \leqslant \sum_{i \in S, j \in T, ij \in E(G)} c(ij) - \sum_{i \in S, j \in T, ji \in E(G)} m(ji)$$

(b) Starting from R we build a new network $\overline{R} = (\overline{G}, \overline{s}, \overline{t}, \overline{c})$, where

- $V(\overline{G}) = V(G) \cup \{\overline{s}, \overline{t}\}$ (these are new vertices);
- $E(\overline{G}) = E(G) \cup \{st, ts\} \cup \{\overline{s}v, v\overline{t} \,:\, v \in V(G)\}$;
- for every $v \in V(G)$, $\overline{c}(\overline{s}v) = \sum_{uv \in E(G)} m(uv)$ and $\overline{c}(v\overline{t}) = \sum_{vu \in E(G)} m(vu)$;
- $\overline{c}(st) = \overline{c}(ts) = \infty$, and, for every $ij \in E(G)$, $\overline{c}(ij) = c(ij) - m(ij)$.

Exercise 8 (cont'd).

Show that there exists a legal flow in R if and only if there exists a flow of value $M = \sum_{e \in E(G)} m(e)$ in $\overline{R}$.

(c) Using a starting *legal flow* (see b)) show how to modify the Ford&Fulkerson algorithm in order to obtain a maximum value legal flow in a network as above.

Exercise 9. Let $R = (G, s, t, c)$ be a flow network.

(a) Design an efficient algorithm that verifies if a given cut (S, T) is of minimum capacity in R.

(b) Design an $\mathcal{O}(n + m)$ time complexity algorithm that verifies if a given flow, x, is of maximum value in R.

(c) Let $c : E \to \mathbb{Z}_+$ and x^* be a maximum flow in R. Suppose that the capacity of a certain arc, $e_0 \in E(G)$, is increased by 1. Devise an efficient algorithm which has to find a maximum flow in the new network.

Exercises for the 9th seminar

Exercise 10. Let x be an integral flow in a given network $R = (G, s, t, c)$, with $v(\mathrm{x}) = v_1 + v_2 + \cdots v_p$, where $v_i \in \mathbb{N}^*$, $\forall i = \overline{1, p}$, $p \geqslant 1$. Prove that there exists p integral flows $\mathrm{x}^1, \mathrm{x}^2, \ldots, \mathrm{x}^p$ such that $v(\mathrm{x}^i) = v_i$, $\forall i = \overline{1, p}$ and

$$\mathrm{x} = \sum_{i=1}^{p} \mathrm{x}^i.$$