

1

Network flows

- Augmenting paths
- Cuts
- Augmenting path theorem
- Integral flow theorem
- Max flow - Min cut theorem
- Ford & Fulkerson algorithm
- Edmonds & Karp algorithm

2

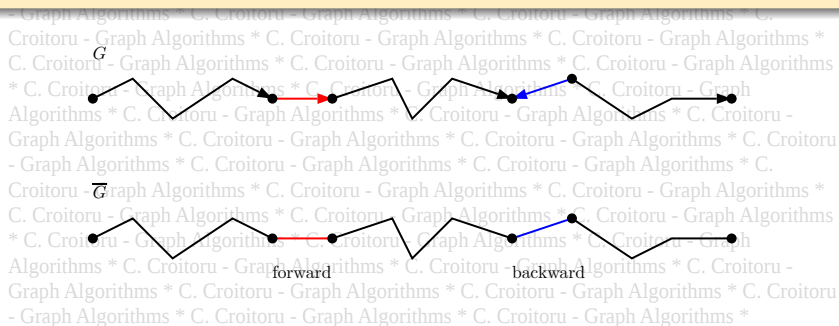
Exercises for the 9th seminar (december 2 - 5 week)

Maximum flow problem - Augmenting paths

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

Definition

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



Maximum flow problem - Augmenting paths

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

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

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

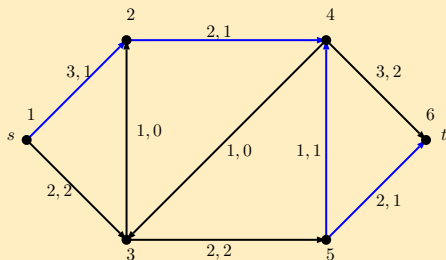
The **residual capacity** of the path P is $r(P) = \min_{e \in E(P)} r(e)$.

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Maximum flow problem - Augmenting paths

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 : $r(P) = \min \{2, 1, 1, 1\} = 1$.

Maximum flow problem - Augmenting paths

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

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 .

* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

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

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

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

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Maximum flow problem - Augmenting paths

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

Proof. By the definition of $r(P)$, constraints (i) hold for x^1 . Constraints (ii) which hold for x - are not changed w. r. t. 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:

$$\begin{aligned} & \sum_j x_{ji}^1 - \sum_j x_{ij}^1 = \sum_{j \neq l} x_{ji} - \sum_{j \neq k} x_{ij} + x_{li}^1 - x_{ik}^1 \\ & = \sum_{j \neq l} x_{ji} - \sum_{j \neq k} x_{ij} + x_{li} + r(P) - x_{ik} - r(P) = \sum_j x_{ji} - \sum_j x_{ij} = 0. \end{aligned}$$

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

$$\begin{aligned} & \sum_j x_{ji}^1 - \sum_j x_{ij}^1 = \sum_{j \neq l, k} x_{ji} - \sum_j x_{ij} + x_{li}^1 + x_{ki}^1 \\ & = \sum_{j \neq l, k} x_{ji} - \sum_j x_{ij} + x_{li} + r(P) + x_{ki} - r(P) = \sum_j x_{ji} - \sum_j x_{ij} = 0. \end{aligned}$$

Maximum flow problem - Augmenting paths

- c) li is a backward arc and ik is a forward arc: similar to b).
d) li and ik are both backward arcs: similar to a).

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

lt is a forward arc:

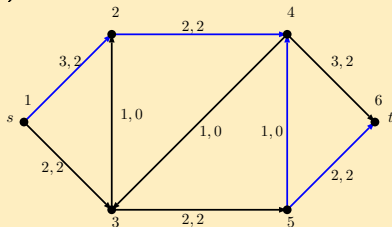
$$\begin{aligned}v(x^1) &= \sum_j x_{jt}^1 - \sum_j x_{tj}^1 = \sum_{j \neq l} x_{jt} - \sum_j x_{tj} + x_{lt}^1 = \\ &= \sum_{j \neq l} x_{jt} - \sum_j x_{tj} + x_{lt} + r(P) = v(x) + r(P).\end{aligned}$$

lt is a backward arc:

$$\begin{aligned}v(x^1) &= \sum_j x_{jt}^1 - \sum_j x_{tj}^1 = \sum_j x_{jt} - \sum_{j \neq l} x_{tj} - x_{tl}^1 = \\ &= \sum_j x_{jt} - \sum_{j \neq l} x_{tj} - (x_{tl} - r(P)) = v(x) + r(P). \quad \square\end{aligned}$$

Maximum flow problem - Augmenting paths

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:



* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

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.**

Maximum flow problem - Cuts

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

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

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

Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Lemma 2

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

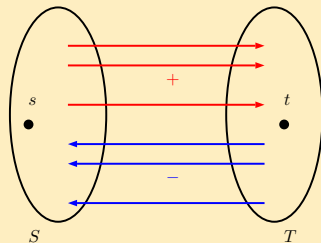
$$v(x) = \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji})$$

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

Maximum flow problem - Cuts

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph



Proof.

$$\begin{aligned} v(x) &= \left(\sum_j x_{sj} - \sum_j x_{js} \right) + 0 = \\ &= \left(\sum_j x_{sj} - \sum_j x_{js} \right) + \sum_{i \in S, i \neq s} \left(\sum_j x_{ij} - \sum_j x_{ji} \right) = \end{aligned}$$

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

Maximum flow problem - Cuts

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

Proof. By Lemma 2

$$\begin{aligned}v(x) &= \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}) \stackrel{x_{ij} \leq c_{ij}}{\leq} \sum_{i \in S} \sum_{j \in T} (c_{ij} - x_{ji}) \stackrel{x_{ji} \geq 0}{\leq} \\ &\leq \sum_{i \in S} \sum_{j \in T} c_{ij} = c(S, T). \quad \square\end{aligned}$$

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

Remark

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

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

Maximum flow problem - Augmenting path theorem

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

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 .

* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

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 .

Maximum flow problem - Integral flow theorem

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

Theorem 2

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

Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

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).

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Maximum flow problem - Max flow - Min cut theorem

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

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 .

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru

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)**.

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Maximum flow problem - Ford & Fulkerson algorithm

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

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.

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Maximum flow problem - Ford & Fulkerson algorithm

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

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, ∞) . The other vertices receive labels by scanning the already labeled vertices:

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Maximum flow problem - Ford & Fulkerson algorithm

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

```
procedure scan(i)
for (j ∈ V, unlabeled) do
  if (ij ∈ 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 ∈ 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:

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

Maximum flow problem - Ford & Fulkerson algorithm

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

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.

- Graph Algorithms - C. Croitoru - Graph Algorithms - C. Croitoru - Graph Algorithms

Maximum flow problem - Ford & Fulkerson algorithm

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

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 $v \leq (n - 1)U$ (this is an upper bound of the capacity of the cut $(\{s\}, V \setminus \{s\})$), therefore the algorithm has $\mathcal{O}(nmU)$ time complexity.

* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

Remark

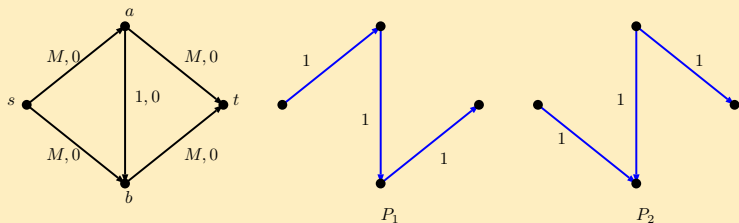
The algorithm may not terminate for irrational capacities. This case does not arise in practical implementations, but the weakness of the algorithm is given by the fact that the number of flow augmentings may depend on the capacities values (and not the size of the network).

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

Maximum flow problem - Ford & Fulkerson algorithm

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

Example



If the **choose** operation in the above algorithm makes that the successive augmenting paths are $P_1, P_2, P_1, P_2, \dots$, 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.

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Edmonds & Karp modification of Ford & Fulkerson algorithm

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

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)).

* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

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 .

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

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

$$x^1 \leftarrow x;$$

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

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Edmonds & Karp modification of Ford & Fulkerson algorithm

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

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

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

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

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C.

Lemma 4

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

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

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru

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, \dots , 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).

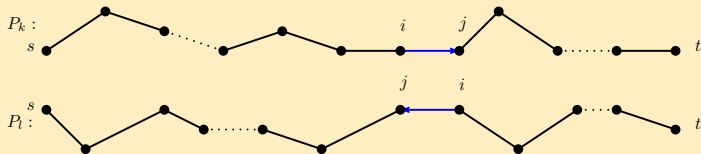
Let ij be a critical arc on the shortest augmenting path P_k w.r.t. x^k . The length of P_k is $\sigma_i^k + \tau_i^k = \sigma_j^k + \tau_j^k$

Edmonds & Karp modification of Ford & Fulkerson algorithm

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

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:

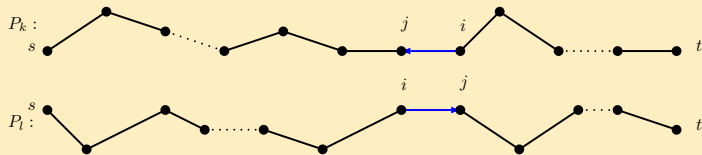
ij is a forward arc in P_k . Then $\sigma_j^k = \sigma_i^k + 1$, in P_l ij will be backward arc, therefore $\sigma_i^l = \sigma_j^l + 1$.



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

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

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).

Edmonds & Karp modification of Ford & Fulkerson algorithm

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

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

Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Corollary

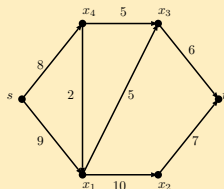
In any flow network there is a flow x with the property that there is no augmenting path w.r.t. x .

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

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 .

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 (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;  
   $K \leftarrow 2^{1 + \lceil \log C \rceil}$ ;  
  while( $K \geq 1$ ) {  
    while( $x$  has an augmenting path  $P$  with  $r(P) \geq K$ )  
       $x \leftarrow x \otimes r(P)$ ;  
     $K \leftarrow K/2$ ;  
  }  
  return  $x$ ;  
}
```

Exercise 2 (cont'd).

1. Prove that procedure $\text{SC-MAX-FLOW}(R)$ 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, $\text{load}(v) \in \mathbb{R}_+$. 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 8. Let $R = (G, s, t, c)$ be a network such that $st, ts \notin E(G)$. Furthermore we have a lower bound function $m : E(G) \rightarrow \mathbb{R}_+$, $m(e) \leq c(e), \forall e \in E(G)$. A flow φ in R is called *legal flow in R* if $x(e) \geq m(e), \forall e \in E(G)$.

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

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

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

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

$$\sum_{vu \in E(G)} m(vu);$$

- $\bar{c}(st) = \bar{c}(ts) = \infty$, and, for every $ij \in E(G)$, $\bar{c}(ij) = c(ij) - m(ij)$.

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 \bar{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 $c : E \rightarrow \mathbb{Z}_+$ 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

