ISyE/Math/CS 425
Introduction to Combinatorial Optimization

3. Maximum Flow Problems

Prof. Carla Michini
University of Wisconsin-Madison

Outline

- We will see the Maximum Flow Problem and the Minimum Cut Problem.
- We will prove the Max-Flow Min-Cut Theorem.
- We will study the Augmenting Path algorithm for solving both problems at once.

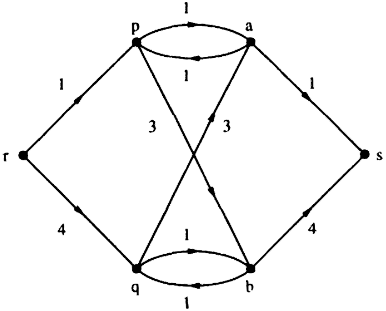
3.2 Maximum Flow Problems

Trucks problem

- We want to send as many trucks as possible from one point r in a street network to another point s.
- For each street segment e, there is an upper bound u_e on the number of trucks that are allowed to use e.

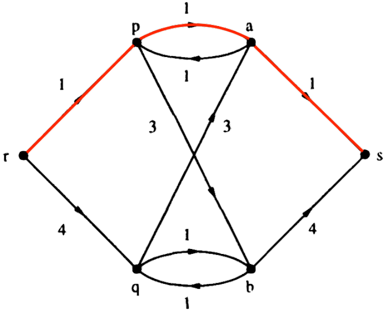
Dipaths in a digraph

- Problem: on a digraph G, find a family $(P_1, \ldots, P_k)$ of (r, s)-dipaths in G, such that each arc e is an arc of at most u_e of the dipaths and such that k is maximized.



Dipaths in a digraph

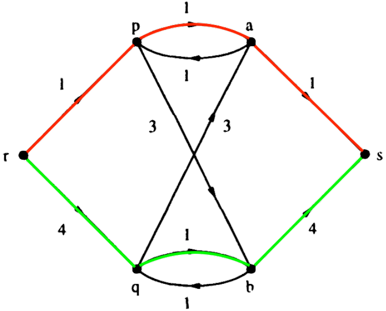
- Problem: on a digraph G, find a family $(P_1, \ldots, P_k)$ of (r, s)-dipaths in G, such that each arc e is an arc of at most u_e of the dipaths and such that k is maximized.



Example: $P_1 = r, p, a, s$.

Dipaths in a digraph

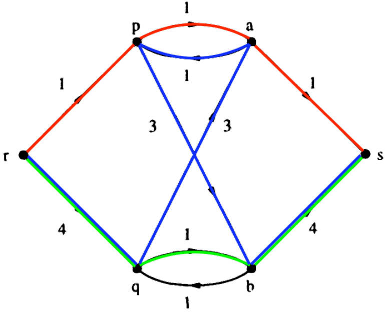
- Problem: on a digraph G, find a family $(P_1, \ldots, P_k)$ of (r, s)-dipaths in G, such that each arc e is an arc of at most u_e of the dipaths and such that k is maximized.



Example: $P_1 = r, p, a, s$, $P_2 = r, q, b, s$.

Dipaths in a digraph

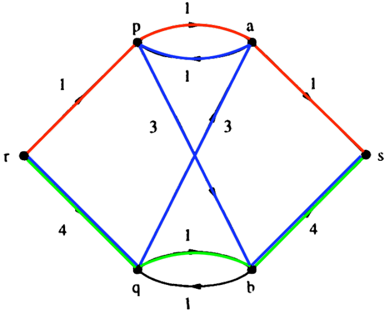
- Problem: on a digraph G, find a family $(P_1, \ldots, P_k)$ of (r, s)-dipaths in G, such that each arc e is an arc of at most u_e of the dipaths and such that k is maximized.

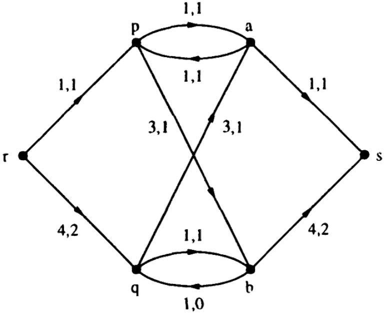


Example: $P_1 = r, p, a, s$, $P_2 = r, q, b, s$ and $P_3 = r, q, a, p, b, s$.

Flows

For every arc $e \in E$, we define $x_e := |\{i : P_i \text{ uses } e\}|$.





Remark 1: we can restrict to simple dipaths and to simple digraphs.
Remark 2: for each $v \in V \setminus \{r, s\}$, P_i must enter and leave v the same number of times.

Flows

$x_e = |\{i : P_i \text{ uses } e\}|$ satisfies:

$$\begin{aligned}
\sum_{wv \in E} x_{wv} - \sum_{vw \in E} x_{vw} &= 0 && \forall v \in V \setminus \{r, s\} \tag{1} \\
0 \leq x_e &\leq u_e && \forall e \in E \tag{2} \\
x_e &\text{ integer} && \forall e \in E \tag{3}
\end{aligned}$$

- r is the source, s is the sink.
- x is an (r, s)-flow if it satisfies (1).
- It is feasible if it also satisfies (2).
- It is integral if it also satisfies (3).

Flows

$$\begin{align}
\sum_{wv \in E} x_{wv} - \sum_{vw \in E} x_{vw} &= 0 & \forall v \in V \setminus \{r, s\} \tag{1}\\
0 \leq x_e &\leq u_e & \forall e \in E \tag{2}\\
x_e \text{ integer} & & \forall e \in E \tag{3}
\end{align}$$

- The net flow into v is

$$f_x(v) := \sum_{wv \in E} x_{wv} - \sum_{vw \in E} x_{vw}.$$

- The value of x is $f_x(s)$.
- The number k of dipaths satisfies $k = f_x(s)$.
- x is acyclic if there is no dicircuit C, each of whose arcs e has $x_e > 0$.

Dipaths & Flows

One can recover dipaths from integral feasible flows:

Proposition 3.1

There exists a family $(P_1, \ldots, P_k)$ of (r, s)-dipaths such that

$$|\{i : P_i \text{ uses } e\}| \leq u_e \qquad \text{for all } e \in E$$

if and only if there exists an integral feasible (r, s)-flow of value k.

Let's prove it!

Maximum Integral Flow Problem

By Proposition 3.1, we can solve our problem if we can solve the following:

Maximum Integral Flow Problem

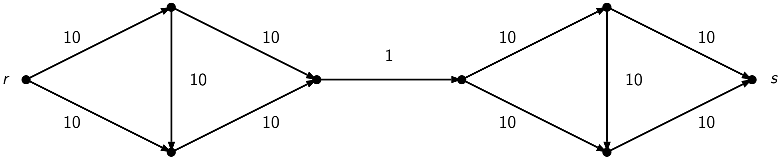
$$\begin{array}{rll} \text{maximize} & f_x(s) & \\ \text{subject to} & f_x(v) = 0 & \forall v \in V \setminus \{r, s\} \\ & 0 \leq x_e \leq u_e & \forall e \in E \\ & x_e \text{ integer} & \forall e \in E \end{array}$$

Maximum Flow problem

- If there is no restriction of integrality, the problem is called the maximum flow problem
- The numbers u_e are called capacities.
  We allow them to be non-negative real numbers or ∞.
  The latter just means that there is no upper bound on x_e.

Flows & cuts

There is a natural way to get upper bounds for the maximum value of a flow.



How?

Flows & cuts

- A cut is a set

  $$\delta(R) := \{vw : vw \in E, v \in R, w \notin R\},$$

  for some $R \subseteq V$.
- The capacity of a cut $\delta(R)$ is $u(\delta(R)) = \sum_{vw \in \delta(R)} u_{vw}$.
- An (r, s)-cut is a cut $\delta(R)$ for which $r \in R$, $s \notin R$.
- For any $A \subseteq V$ we use $\bar{A}$ to denote $V \setminus A$.

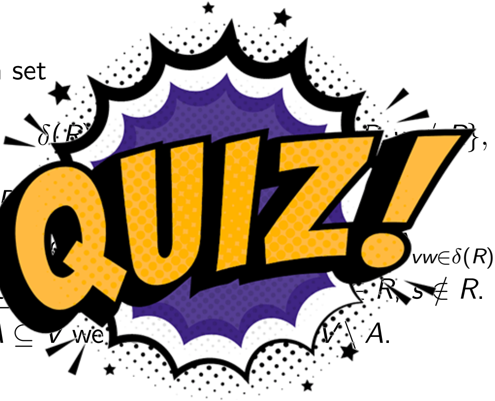
Flows & cuts

- A cut is a set

$$\delta(R \qquad R, w \notin R\},$$

  for some R

- The capa $_{vw\in\delta(R)} u_{vw}$.
- An (r, s)-c $\in R, s \notin R$.
- For any $A \subseteq V$ we $V \setminus A$.



Max-Flow Min-Cut Theorem

We now show how to upper bound the maximum flow value.

Proposition 3.3

For any (r, s)-cut $\delta(R)$ and any (r, s)-flow x, we have

$$x(\delta(R)) - x(\delta(\bar{R})) = f_x(s).$$

Proof: see Notes.

Max-Flow Min-Cut Theorem

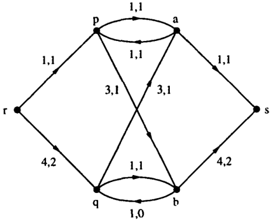
We now show how to upper bound the maximum flow value.

Proposition 3.3

For any (r, s)-cut $\delta(R)$ and any (r, s)-flow x, we have

$$x(\delta(R)) - x(\delta(\bar{R})) = f_x(s).$$

Proof: see Notes.



$R = \{r, p, q\}$ gives $x(\delta(R)) - x(\delta(\bar{R})) = 4 - 1 = 3$.

Max-Flow Min-Cut Theorem

We now show how to upper bound the maximum flow value.

Proposition 3.3

For any (r, s)-cut $\delta(R)$ and any (r, s)-flow x, we have

$$x(\delta(R)) - x(\delta(\bar{R})) = f_x(s).$$

Corollary 3.4

For any feasible (r, s)-flow x and any (r, s)-cut $\delta(R)$, we have

$$f_x(s) \leq u(\delta(R))$$

Proof: see Notes.

Max-Flow Min-Cut Theorem

Corollary 3.4

For any feasible (r, s)-flow x and any (r, s)-cut $\delta(R)$, we have

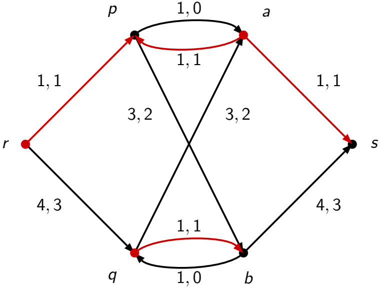
$$f_x(s) \leq u(\delta(R))$$

- Therefore, the maximum flow value is bounded by the minimum cut capacity.
- So if we can find a flow and a cut such that the value of the flow is equal to the capacity of the cut then we know that the flow is maximum.
- We will show that this can always be done!

Max-Flow Min-Cut Theorem

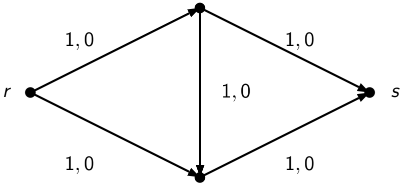
Theorem 3.5 (Max-Flow Min-Cut Theorem)
If there is a maximum (r, s)-flow, then

$$\max\{f_x(s) : x \text{ feasible } (r, s)\text{-flow}\} =$$
$$= \min\{u(\delta(R)) : \delta(R) \text{ an } (r, s)\text{-cut}\}.$$



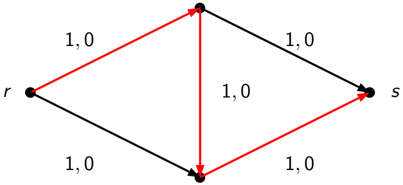
Max-Flow Min-Cut Theorem

- We prove the theorem by solving both optimization problems.
- How can we solve the maximum flow problem?
- Given a feasible flow x, we find one of larger value.
- Idea: We find an (r, s)-dipath P for which $x_e < u_e$ for each arc e of P, then we increase x by the same value on all arcs of P.



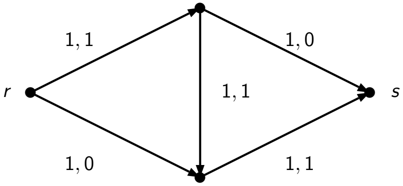
Max-Flow Min-Cut Theorem

- We prove the theorem by solving both optimization problems.
- How can we solve the maximum flow problem?
- Given a feasible flow x, we find one of larger value.
- Idea: We find an (r, s)-dipath P for which $x_e < u_e$ for each arc e of P, then we increase x by the same value on all arcs of P.



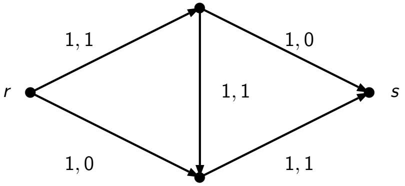
Max-Flow Min-Cut Theorem

- We prove the theorem by solving both optimization problems.
- How can we solve the maximum flow problem?
- Given a feasible flow x, we find one of larger value.
- Idea: We find an (r, s)-dipath P for which $x_e < u_e$ for each arc e of P, then we increase x by the same value on all arcs of P.



Max-Flow Min-Cut Theorem

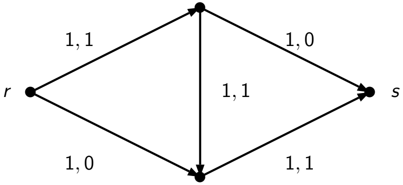
- We prove the theorem by solving both optimization problems.
- How can we solve the maximum flow problem?
- Given a feasible flow x, we find one of larger value.
- Idea: We find an (r, s)-dipath P for which $x_e < u_e$ for each arc e of P, then we increase x by the same value on all arcs of P.



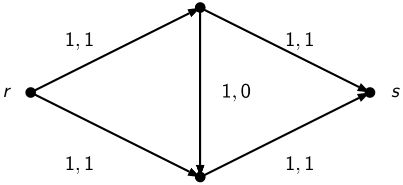
- There are no more such dipaths, but the flow is not maximum!
- This idea is not enough to solve the problem.

Max-Flow Min-Cut Theorem

What we have:

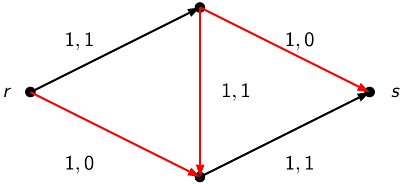


What we want:

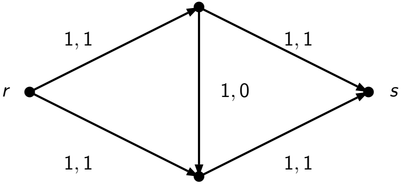


Max-Flow Min-Cut Theorem

What we have:



What we want:



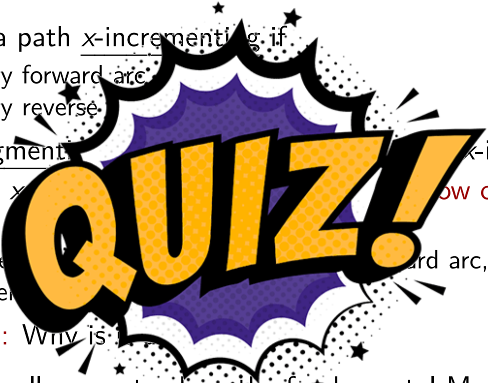
Max-Flow Min-Cut Theorem

- We call a path x-incrementing if
  - Every forward arc e has $x_e < u_e$,
  - Every reverse arc e has $x_e > 0$.
- An x-augmenting path is an (r, s)-path that is x-incrementing.
- Given an x-augmenting path we can find a flow of larger value:
  - Raise x_e by some positive ϵ on each forward arc,
  - Lower x_e by ϵ on each reverse arc.
- Question: Why is it a flow?

  This idea allows us to show the fundamental Max-Flow Min-Cut Theorem...

Max-Flow Min-Cut Theorem

- We call a path *x*-incrementing if
  - Every forward arc
  - Every reverse
- An *x*-augmenti … *x*-incrementing.
- Given an *x* … ow of larger value:
  - Raise … rd arc,
  - Lowe
- Question: Why is



This idea allows us to show the fundamental Max-Flow Min-Cut Theorem...

Max-Flow Min-Cut Theorem

Theorem 3.5 (Max-Flow Min-Cut Theorem)

If there is a maximum (r, s)-flow, then

$$\max\{f_x(s) : x \text{ feasible } (r, s)\text{-flow}\} =$$
$$= \min\{u(\delta(R)) : \delta(R) \text{ an } (r, s)\text{-cut}\}.$$

Proof: see Notes.

Max-Flow Min-Cut Theorem

Theorem 3.5 (Max-Flow Min-Cut Theorem)
If there is a maximum (r, s)-flow, then

$$\max\{f_x(s) : x \text{ feasible } (r,s)\text{-flow}\} =$$
$$= \min\{u(\delta(R)) : \delta(R) \text{ an } (r,s)\text{-cut}\}.$$

Proof: see Notes.

Theorem 3.6
A feasible flow x is maximum if and only ifthere is no x-augmenting path.

Proof: see Notes.

The Augmenting Path Algorithm

- Beginning with $x = 0$, repeatedly find an x-augmenting path P and augment x by the maximum value permitted.
- This value is $\epsilon := \min(\epsilon_1, \epsilon_2)$, where

  $$\begin{aligned} \epsilon_1 &:= \min\{u_e - x_e : e \text{ forward in } P\}, \\ \epsilon_2 &:= \min\{x_e : e \text{ reverse in } P\}. \end{aligned}$$

  We call ϵ the $\underline{x\text{-width}}$ of P.
- If such a path with x-width ∞ is found, then there is no maximum flow, and the algorithm terminates.
- If there is no x-augmenting path, then by Theorem 3.6 x is a maximum flow and the set R of nodes reachable by an x-incrementing path from r determines a minimum cut; again, the algorithm terminates.

The Augmenting Path Algorithm for Integral Flows

- In the Maximum Integral Flow Problem, we have the further constraint that the flow x must be integral.
- In this case, we can replace capacities u_e with the integral $\lfloor u_e \rfloor$.
- In the Augmenting Path Algorithm, in each iteration the flow x remains integral.
- Therefore the maximum flow found will be integral.

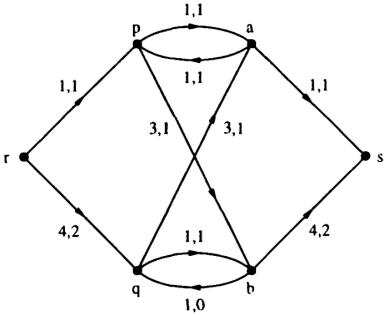
Searching for augmenting paths

- How do we search for augmenting paths?
- Define an auxiliary digraph $\underline{G(x)}$, depending on G, u and the current flow x, as follows.

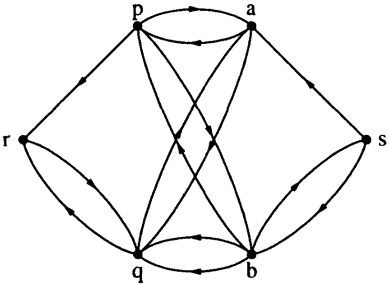
$$\begin{aligned} V(G(x)) &= V, \\ E(G(x)) &= \{vw : vw \in E \text{ and } x_{vw} < u_{vw} \text{ or} \\ &\qquad\qquad wv \in E \text{ and } x_{wv} > 0\}. \end{aligned}$$

Example of auxiliary digraph

Original digraph G with u, x



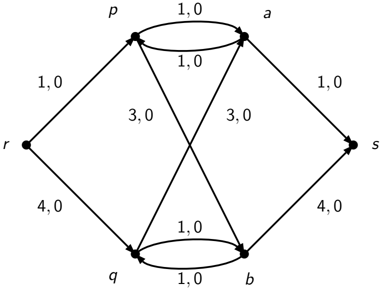
Auxiliary digraph $G(x)$



- x-augmenting paths correspond to (r, s)-dipaths in $G(x)$.
- Each iteration of the maximum flow algorithm can be performed in $O(m)$ time using breadth-first search.

Example of the Augmenting Path Algorithm

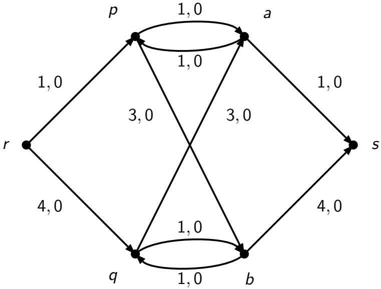
Digraph G with flow



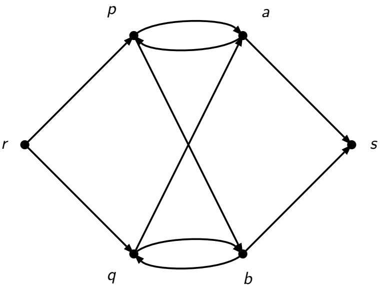
Auxiliary digraph

Example of the Augmenting Path Algorithm

Digraph G with flow

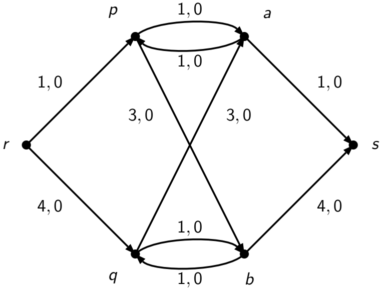


Auxiliary digraph

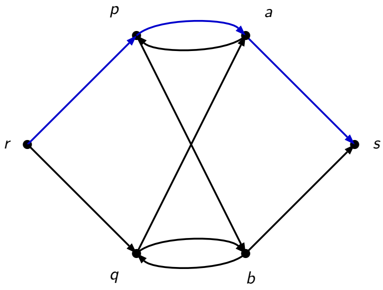


Example of the Augmenting Path Algorithm

Digraph G with flow

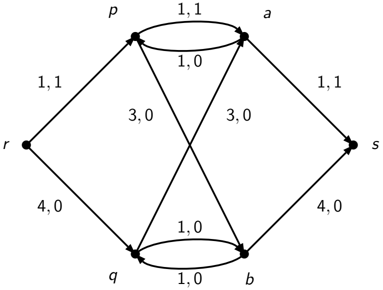


Auxiliary digraph



Example of the Augmenting Path Algorithm

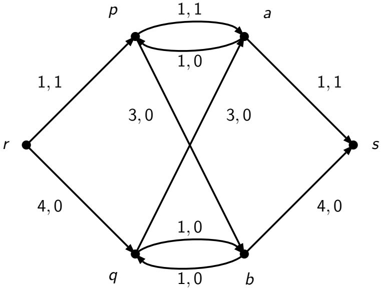
Digraph G with flow



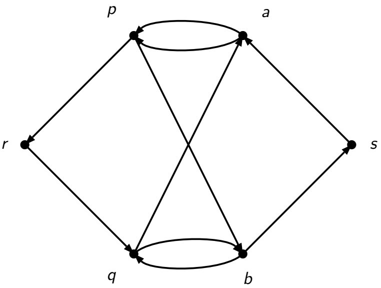
Auxiliary digraph

Example of the Augmenting Path Algorithm

Digraph G with flow

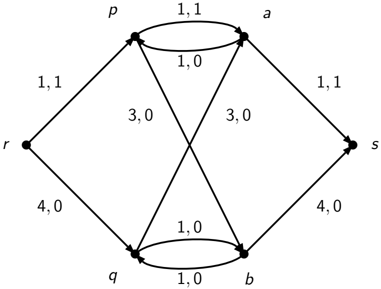


Auxiliary digraph

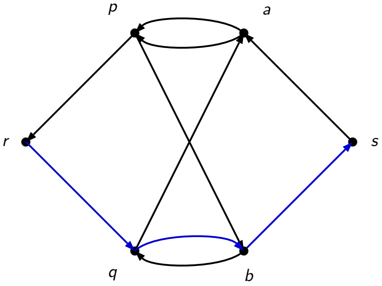


Example of the Augmenting Path Algorithm

Digraph G with flow

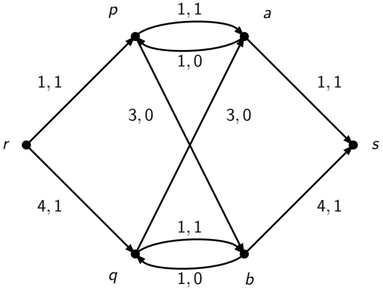


Auxiliary digraph



Example of the Augmenting Path Algorithm

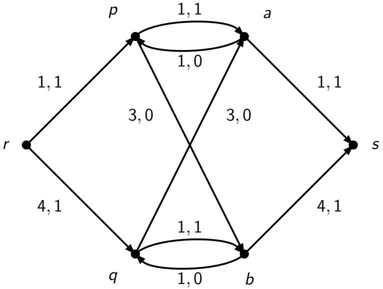
Digraph G with flow



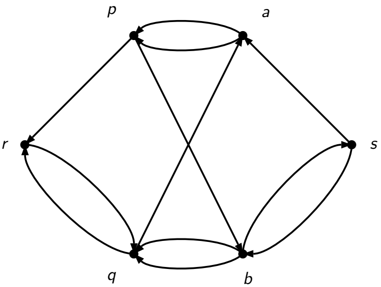
Auxiliary digraph

Example of the Augmenting Path Algorithm

Digraph G with flow

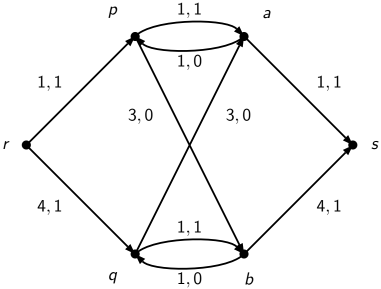


Auxiliary digraph

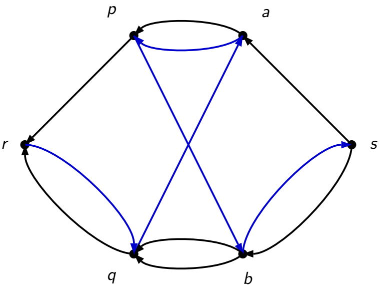


Example of the Augmenting Path Algorithm

Digraph G with flow

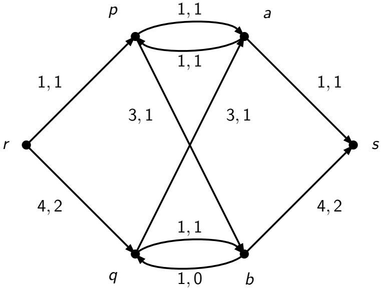


Auxiliary digraph



Example of the Augmenting Path Algorithm

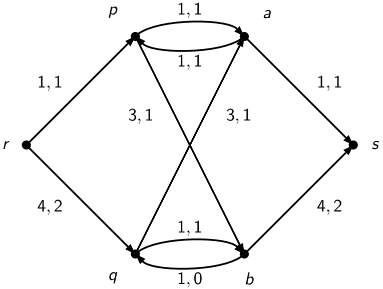
Digraph G with flow



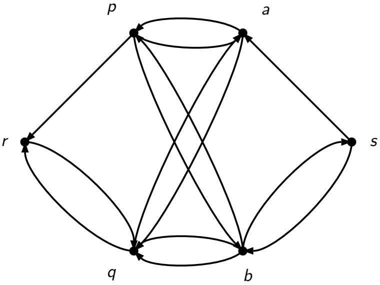
Auxiliary digraph

Example of the Augmenting Path Algorithm

Digraph G with flow

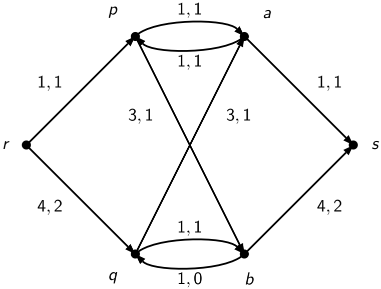


Auxiliary digraph

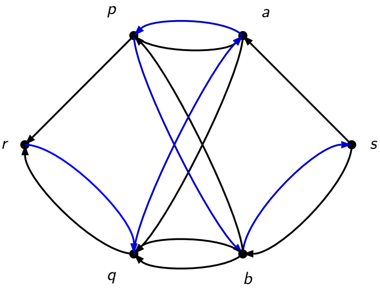


Example of the Augmenting Path Algorithm

Digraph G with flow

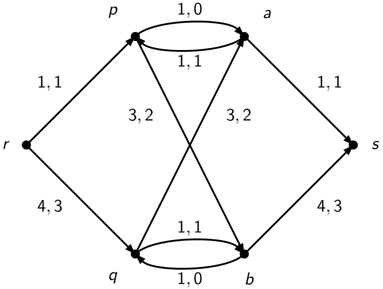


Auxiliary digraph



Example of the Augmenting Path Algorithm

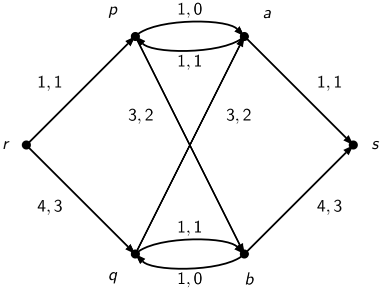
Digraph G with flow



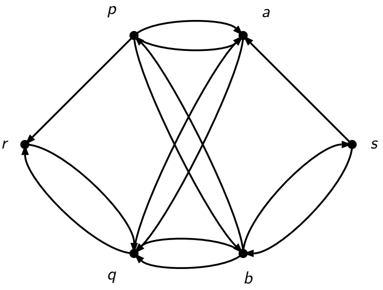
Auxiliary digraph

Example of the Augmenting Path Algorithm

Digraph G with flow

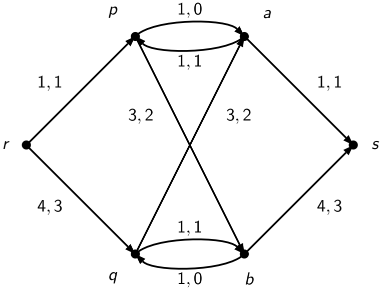


Auxiliary digraph

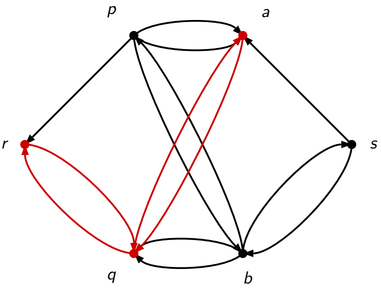


Example of the Augmenting Path Algorithm

Digraph G with flow

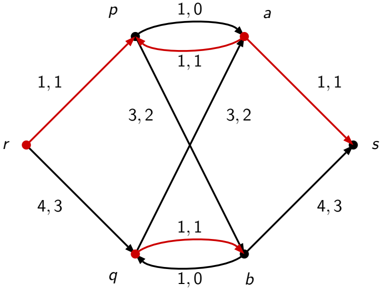


Auxiliary digraph



Example of the Augmenting Path Algorithm

Digraph G with flow



Auxiliary digraph

Number of augmentations

The running time of the algorithm depends directly on the number of augmentations required.

Theorem 3.9

If each component of u is either integral or ∞, and the maximum flow value is $K < \infty$, then the maximum flow algorithm terminates after at most K augmentations.

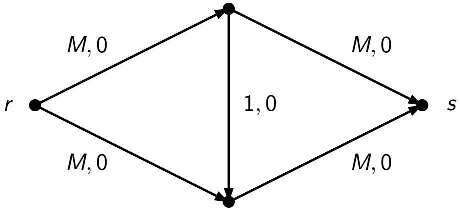
Proof.

- When u is integral, x remains integral, and each augmentation is of value at least 1. □

Number of augmentations

The bound of Theorem 3.9 can actually be attained.

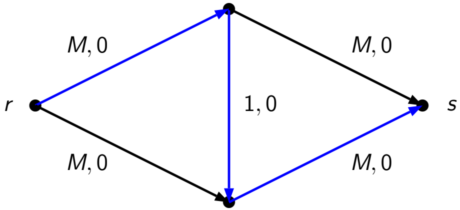
Example:



Number of augmentations

The bound of Theorem 3.9 can actually be attained.

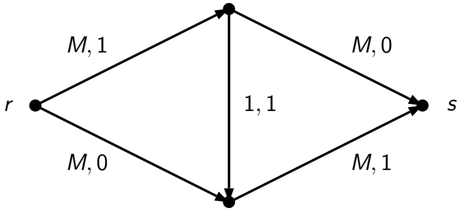
Example:



Number of augmentations

The bound of Theorem 3.9 can actually be attained.

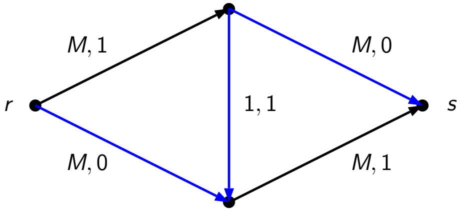
Example:



Number of augmentations

The bound of Theorem 3.9 can actually be attained.

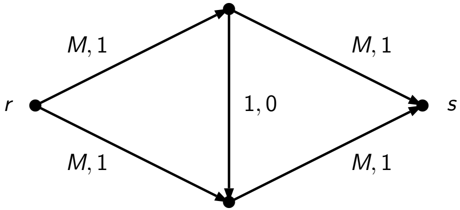
Example:



Number of augmentations

The bound of Theorem 3.9 can actually be attained.

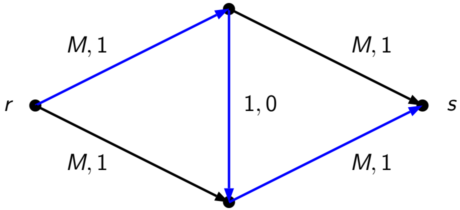
Example:



Number of augmentations

The bound of Theorem 3.9 can actually be attained.

Example:



Shortest Augmenting Paths

- We will now see a bound on the number of augmentations that does not depend on the capacities.
- We call an x-augmenting path shortest if it has the minimum possible number of arcs.

Theorem 3.10

If each augmentation of the augmenting path algorithm is on a shortest augmenting path, then there are at most nm augmentations.

Shortest Augmenting Paths

Corollary 3.11
The augmenting path algorithm with breadth-first search solves the maximum flow problem in time $O(nm^2)$.

Proof.

- By Theorem 3.10 there are at most nm augmentations.
- Breadth-first search finds a shortest augmenting path in time $O(m)$. □

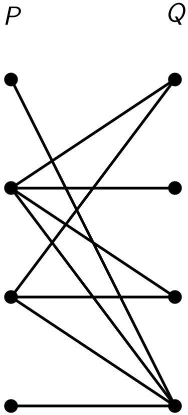
3.3 Applications of Maximum Flow and Minimum Cut

Bipartite Matchings and Covers

- We are given two sets P and Q of people, and the pairs (p, q) that like each other.
- The marriage problem is to arrange as many (monogamous) marriages as possible with the restriction that married people should like each other.

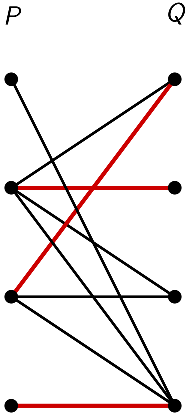
Bipartite graphs

- We can associate with the input a graph $G = (V, E)$ such that $V = P \cup Q$ and $E \subseteq \{pq : p \in P,\ q \in Q\}$.
- Such graphs, where there is a partition of the nodes into two sets such that every edge has its ends in different sets, are called <u>bipartite</u>.
- $\{P, Q\}$ is called a <u>bipartition</u> of G.



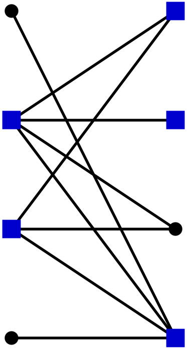
Matchings

- A matching of G is a subset M of E such that no two edges in M share an end.
- The marriage problem asks for a matching of G of maximum size.



Covers

- A cover of a graph G is a set C of nodes such that every edge of G has at least one end in C.



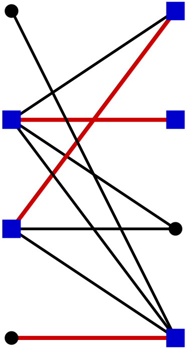
Covers

Observation

For any matching M and any cover C, we have $|M| \leq |C|$.

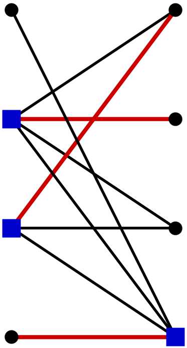
Proof.

- For each edge $vw \in M$, at least one of its ends is in C.
- Because matching edges cannot have an end in common, the corresponding nodes of C are all distinct.
- Therefore, $|M| \leq |C|$. □



Covers

- It follows that, if we can find a matching M and a cover C with $|M| = |C|$, then we know that M is maximum.



König's Theorem

Theorem 3.14 (König's Theorem)

For a bipartite graph G,

$$\max\{|M| : M \text{ a matching}\} = \min\{|C| : C \text{ a cover}\}.$$

Let's prove it using maximum flows!

Proof of König's Theorem

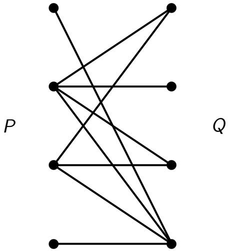
We show how:

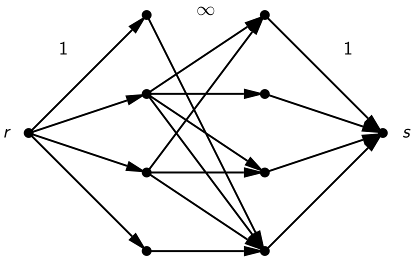
- The Max-Flow Min-Cut Theorem implies König's Theorem.
- A maximum flow algorithm provides an efficient algorithm for constructing a maximum matching and a minimum cover.

Given G with bipartition $\{P, Q\}$, we form a digraph $G' = (V', E')$ with capacity vector u as follows.

Proof of König's Theorem

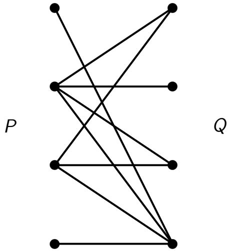
- $V' = V \cup \{r, s\}$, where r, s are new nodes.
- For each edge pq of G with $p \in P$, $q \in Q$, there is an arc $pq \in E'$ with capacity ∞.
- For each $p \in P$ there is an arc $rp \in E'$ of capacity 1.
- For each $q \in Q$ there is an arc $qs \in E'$ of capacity 1.

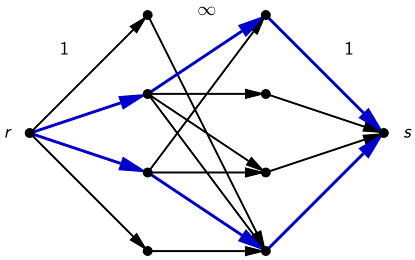




Proof of König's Theorem

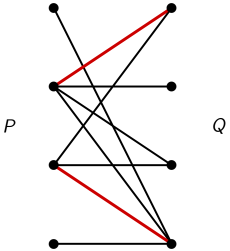
- Let x be an integral feasible flow in G' of value k.
- This implies that x is $\{0,1\}$-valued. Why?
- Define $M \subseteq E$ by: $pq \in M$ if $x_{pq} = 1$ and $pq \notin M$ if $x_{pq} = 0$.
- Then M is a matching of G. Why?
- $|M| = k$ by applying Proposition 3.3 to (r,s)-cut $\delta(\{r\} \cup P)$.

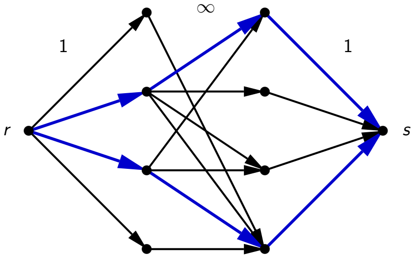




Proof of König's Theorem

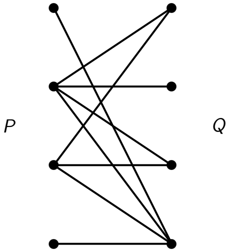
- Let x be an integral feasible flow in G' of value k.
- This implies that x is $\{0, 1\}$-valued. Why?
- Define $M \subseteq E$ by: $pq \in M$ if $x_{pq} = 1$ and $pq \notin M$ if $x_{pq} = 0$.
- Then M is a matching of G. Why?
- $|M| = k$ by applying Proposition 3.3 to (r, s)-cut $\delta(\{r\} \cup P)$.

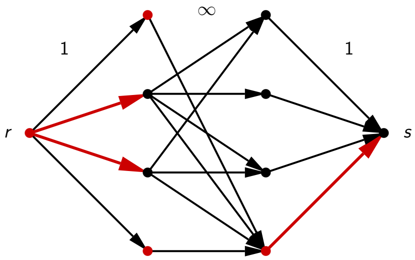




Proof of König's Theorem

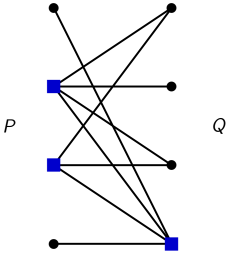
- Now consider a minimum cut $\delta'(\{r\} \cup U)$ where $U \subseteq V$.
- Since it has finite capacity (why?), there can be no edge of G from $P \cap U$ to $Q \setminus U$.
- Therefore, every edge of G is incident with an element of $C := (P \setminus U) \cup (Q \cap U)$. That is, C is a cover.
- The capacity of the cut is $|P \setminus U| + |Q \cap U| = |C|$, so C is a cover of cardinality equal to the max size of a matching.

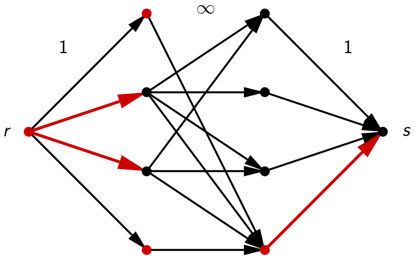




Proof of König's Theorem

- Now consider a minimum cut $\delta'(\{r\} \cup U)$ where $U \subseteq V$.
- Since it has finite capacity (why?), there can be no edge of G from $P \cap U$ to $Q \setminus U$.
- Therefore, every edge of G is incident with an element of $C := (P \setminus U) \cup (Q \cap U)$. That is, C is a cover.
- The capacity of the cut is $|P \setminus U| + |Q \cap U| = |C|$, so C is a cover of cardinality equal to the max size of a matching.





Proof of König's Theorem

- If the maximum integral (r, s)-flow on G' has value k, we can construct a matching M^* of size k.
- By the Max-Flow Min-Cut theorem, the capacity of the minimum (r, s)-cut in G' is k.
- Given a minimum (r, s)-cut in G' of cardinality k, we can construct a cover C^* of cardinality k.
- Since $|M| \leq |C|$ for any matching M and for any cover C of G, we conclude that M^* is a maximum matching and that C^* is a minimum cover, and they have the same size □

Proof of König's Theorem

- Hence we can find a maximum cardinality matching in G by solving the maximum flow problem on G'.
- There will be at most $|P| \leq n$ augmentations, by Theorem 3.9, since the maximum matching size is at most $|P|$.
- So we get an algorithm for maximum bipartite matching having running time $O(mn)$.