Variations on Max-Flow Problems

Math 482, Lecture 27

Misha Lavrov

April 8, 2020

Plan for the lecture

The goal of today is to talk about several problems which can be reduced to max-flow problems (and solved using max-flow algorithms).

Plan for the lecture

The goal of today is to talk about several problems which can be reduced to max-flow problems (and solved using max-flow algorithms).

Problems we'll consider:

- Supply and demand problems.

Plan for the lecture

The goal of today is to talk about several problems which can be reduced to max-flow problems (and solved using max-flow algorithms).

Problems we'll consider:

- Supply and demand problems.
- Finding feasible flows with lower and upper bounds.

Plan for the lecture

The goal of today is to talk about several problems which can be reduced to max-flow problems (and solved using max-flow algorithms).

Problems we'll consider:

- Supply and demand problems.
- Finding feasible flows with lower and upper bounds.
- Bipartite matchings and vertex covers (again).

Posing a supply/demand problem

Recall: given a network (N, A), a flow $\mathbf{x}$, and a node $k \in N$, the *excess at k* is

$$\Delta_k(\mathbf{x}) := \sum_{i:(i,k)\in A} x_{ik} - \sum_{j:(k,j)\in A} x_{kj}.$$

Posing a supply/demand problem

Recall: given a network (N, A), a flow $\mathbf{x}$, and a node $k \in N$, the *excess at k* is

$$\Delta_k(\mathbf{x}) := \sum_{i:(i,k)\in A} x_{ik} - \sum_{j:(k,j)\in A} x_{kj}.$$

In a network flow problem, we want $\Delta_k(\mathbf{x}) = 0$ for $k \neq s, t$. We want to maximize $\Delta_t(\mathbf{x})$.

Posing a supply/demand problem

Recall: given a network (N, A), a flow $\mathbf{x}$, and a node $k \in N$, the *excess at k* is

$$\Delta_k(\mathbf{x}) := \sum_{i:(i,k)\in A} x_{ik} - \sum_{j:(k,j)\in A} x_{kj}.$$

In a network flow problem, we want $\Delta_k(\mathbf{x}) = 0$ for $k \neq s, t$. We want to maximize $\Delta_t(\mathbf{x})$.

In a supply/demand problem, there is no source or sink. We have a vector $\mathbf{d}$ of demands. For *every* node k, we want $\Delta_k(\mathbf{x}) = d_k$.

Posing a supply/demand problem

Recall: given a network (N, A), a flow $\mathbf{x}$, and a node $k \in N$, the *excess at* k is

$$\Delta_k(\mathbf{x}) := \sum_{i:(i,k)\in A} x_{ik} - \sum_{j:(k,j)\in A} x_{kj}.$$

In a network flow problem, we want $\Delta_k(\mathbf{x}) = 0$ for $k \neq s, t$. We want to maximize $\Delta_t(\mathbf{x})$.

In a supply/demand problem, there is no source or sink. We have a vector $\mathbf{d}$ of demands. For *every* node k, we want $\Delta_k(\mathbf{x}) = d_k$.

- When $d_k < 0$, k is a *supply node*: it has extra flow it wants to get rid of.
- When $d_k > 0$, k is a *demand node*: it wants more entering than leaving flow.

Properties of supply/demand problems

Observations:

- The supply/demand problem is a feasibility problem: we have no objective function, we just want to see if it's possible to satisfy all demands.

Properties of supply/demand problems

Observations:

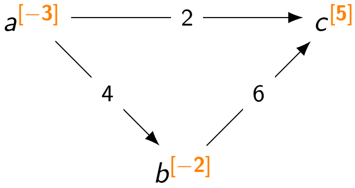
- The supply/demand problem is a feasibility problem: we have no objective function, we just want to see if it's possible to satisfy all demands.
- We know it's impossible if $\sum_{k \in N} d_k \neq 0$. Total supply must equal total demand!

Properties of supply/demand problems

Observations:

- The supply/demand problem is a feasibility problem: we have no objective function, we just want to see if it's possible to satisfy all demands.
- We know it's impossible if $\sum_{k \in N} d_k \neq 0$. Total supply must equal total demand!

Example problem:



Reducing supply/demand problems to max-flow

We solve supply/demand problems by turning them into an equivalent max-flow problem. Then, we solve the max-flow problem and undo the transformation.

Reducing supply/demand problems to max-flow

We solve supply/demand problems by turning them into an equivalent max-flow problem. Then, we solve the max-flow problem and undo the transformation.

Rule for constructing the max-flow instance:

1. Add new nodes s and t.

Reducing supply/demand problems to max-flow

We solve supply/demand problems by turning them into an equivalent max-flow problem. Then, we solve the max-flow problem and undo the transformation.

Rule for constructing the max-flow instance:

1. Add new nodes s and t.
2. For every k with $d_k > 0$, add an arc (k, t) with capacity d_k.

Reducing supply/demand problems to max-flow

We solve supply/demand problems by turning them into an equivalent max-flow problem. Then, we solve the max-flow problem and undo the transformation.

Rule for constructing the max-flow instance:

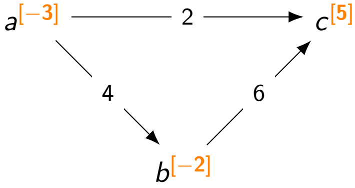
1. Add new nodes s and t.
2. For every k with $d_k > 0$, add an arc (k, t) with capacity d_k.
3. For every k with $d_k < 0$, add an arc (s, k) with capacity $-d_k$.

Reducing supply/demand problems to max-flow

We solve supply/demand problems by turning them into an equivalent max-flow problem. Then, we solve the max-flow problem and undo the transformation.

Rule for constructing the max-flow instance:

1. Add new nodes s and t.
2. For every k with $d_k > 0$, add an arc (k, t) with capacity d_k.
3. For every k with $d_k < 0$, add an arc (s, k) with capacity $-d_k$.

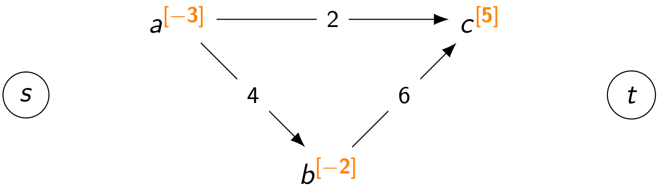


Reducing supply/demand problems to max-flow

We solve supply/demand problems by turning them into an equivalent max-flow problem. Then, we solve the max-flow problem and undo the transformation.

Rule for constructing the max-flow instance:

1. Add new nodes s and t.
2. For every k with $d_k > 0$, add an arc (k, t) with capacity d_k.
3. For every k with $d_k < 0$, add an arc (s, k) with capacity $-d_k$.

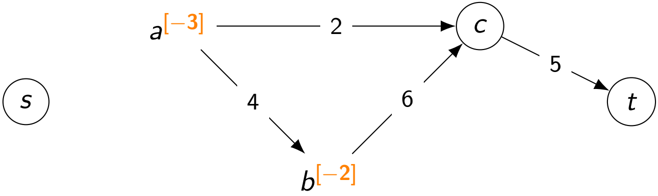


Reducing supply/demand problems to max-flow

We solve supply/demand problems by turning them into an equivalent max-flow problem. Then, we solve the max-flow problem and undo the transformation.

Rule for constructing the max-flow instance:

1. Add new nodes s and t.
2. For every k with $d_k > 0$, add an arc (k, t) with capacity d_k.
3. For every k with $d_k < 0$, add an arc (s, k) with capacity $-d_k$.

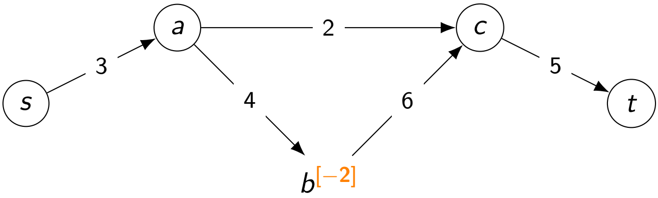


Reducing supply/demand problems to max-flow

We solve supply/demand problems by turning them into an equivalent max-flow problem. Then, we solve the max-flow problem and undo the transformation.

Rule for constructing the max-flow instance:

1. Add new nodes s and t.
2. For every k with $d_k > 0$, add an arc (k, t) with capacity d_k.
3. For every k with $d_k < 0$, add an arc (s, k) with capacity $-d_k$.

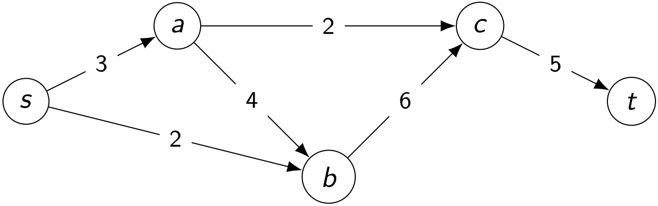


Reducing supply/demand problems to max-flow

We solve supply/demand problems by turning them into an equivalent max-flow problem. Then, we solve the max-flow problem and undo the transformation.

Rule for constructing the max-flow instance:

1. Add new nodes s and t.
2. For every k with $d_k > 0$, add an arc (k, t) with capacity d_k.
3. For every k with $d_k < 0$, add an arc (s, k) with capacity $-d_k$.



Going from max-flow back to supply/demand

The supply/demand problem is only feasible if all the arcs into t are at capacity in the maximum flow. (Then, all arcs out of s will also be at capacity.)

Going from max-flow back to supply/demand

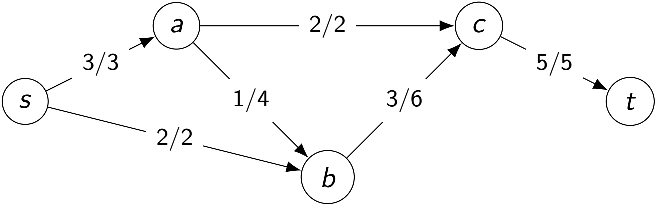
The supply/demand problem is only feasible if all the arcs into t are at capacity in the maximum flow. (Then, all arcs out of s will also be at capacity.)

If this happens, we get the supply/demand solution by erasing nodes s, t and all their arcs.

Going from max-flow back to supply/demand

The supply/demand problem is only feasible if all the arcs into t are at capacity in the maximum flow. (Then, all arcs out of s will also be at capacity.)

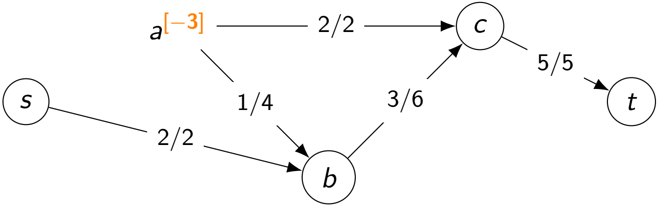
If this happens, we get the supply/demand solution by erasing nodes s, t and all their arcs.



Going from max-flow back to supply/demand

The supply/demand problem is only feasible if all the arcs into t are at capacity in the maximum flow. (Then, all arcs out of s will also be at capacity.)

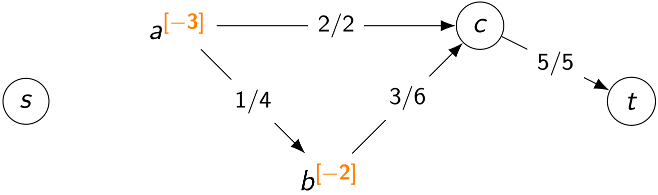
If this happens, we get the supply/demand solution by erasing nodes s, t and all their arcs.



Going from max-flow back to supply/demand

The supply/demand problem is only feasible if all the arcs into t are at capacity in the maximum flow. (Then, all arcs out of s will also be at capacity.)

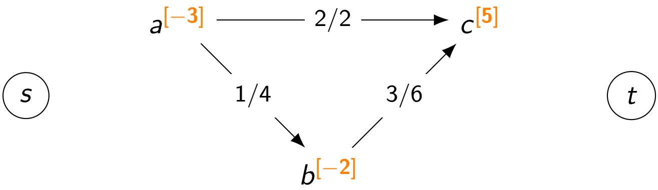
If this happens, we get the supply/demand solution by erasing nodes s, t and all their arcs.



Going from max-flow back to supply/demand

The supply/demand problem is only feasible if all the arcs into t are at capacity in the maximum flow. (Then, all arcs out of s will also be at capacity.)

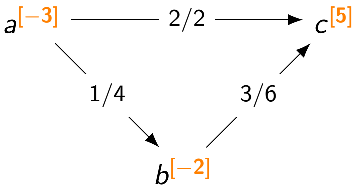
If this happens, we get the supply/demand solution by erasing nodes s, t and all their arcs.



Going from max-flow back to supply/demand

The supply/demand problem is only feasible if all the arcs into t are at capacity in the maximum flow. (Then, all arcs out of s will also be at capacity.)

If this happens, we get the supply/demand solution by erasing nodes s, t and all their arcs.



Feasible circulation

The feasible circulation problem is, again, a feasibility problem in a network with no source or sink.

Feasible circulation

The feasible circulation problem is, again, a feasibility problem in a network with no source or sink.

- We want flow conservation to hold at every node.

Feasible circulation

The feasible circulation problem is, again, a feasibility problem in a network with no source or sink.

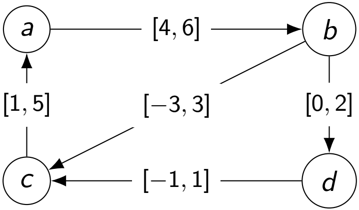
- We want flow conservation to hold at every node.
- However, the arcs now have lower and upper bounds: for each arc (i, j), we're given an interval $[a_{ij}, b_{ij}]$ and ask that $a_{ij} \leq x_{ij} \leq b_{ij}$.

Feasible circulation

The feasible circulation problem is, again, a feasibility problem in a network with no source or sink.

- We want flow conservation to hold at every node.
- However, the arcs now have lower and upper bounds: for each arc (i, j), we're given an interval $[a_{ij}, b_{ij}]$ and ask that $a_{ij} \leq x_{ij} \leq b_{ij}$.

Example problem:



Reducing feasible circulation to supply/demand

We solve feasible circulation problems by turning them into an equivalent supply/demand problem (which we now understand).

Reducing feasible circulation to supply/demand

We solve feasible circulation problems by turning them into an equivalent supply/demand problem (which we now understand).

Rules for constructing the supply/demand problem:

1. Start by setting $d_k = 0$ for all k.

Reducing feasible circulation to supply/demand

We solve feasible circulation problems by turning them into an equivalent supply/demand problem (which we now understand).

Rules for constructing the supply/demand problem:

1. Start by setting $d_k = 0$ for all k.
2. For every arc (i, j) with interval $[a_{ij}, b_{ij}]$, instead give it $c_{ij} = b_{ij} - a_{ij} \ldots$
3. $\ldots$ but add a_{ij} to d_i, and subtract a_{ij} from d_j.

Reducing feasible circulation to supply/demand

We solve feasible circulation problems by turning them into an equivalent supply/demand problem (which we now understand).

Rules for constructing the supply/demand problem:

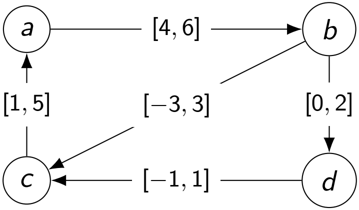
1. Start by setting $d_k = 0$ for all k.
2. For every arc (i, j) with interval $[a_{ij}, b_{ij}]$, instead give it $c_{ij} = b_{ij} - a_{ij} \ldots$
3. $\ldots$ but add a_{ij} to d_i, and subtract a_{ij} from d_j.
4. In the end, each node k has

$$d_k = \sum_{j:(k,j)\in A} a_{kj} - \sum_{i:(i,k)\in A} a_{ik}.$$

(We could skip directly to this step.)

Example of the reduction

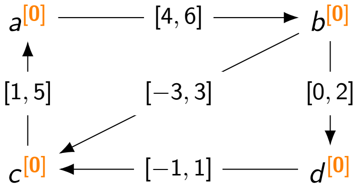
Here is an example:



We'd solve the resulting supply/demand problem by adding a source s, a sink t, and arcs from s or to t, as we discussed earlier.

Example of the reduction

Here is an example:

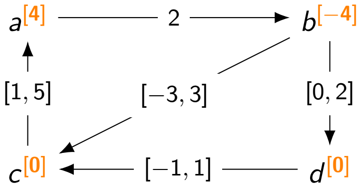


We'd solve the resulting supply/demand problem by adding a source s, a sink t, and arcs from s or to t, as we discussed earlier.

Then, we need to convert the resulting supply/demand solution to a feasible circulation.

Example of the reduction

Here is an example:



We'd solve the resulting supply/demand problem by adding a source s, a sink t, and arcs from s or to t, as we discussed earlier.

Then, we need to convert the resulting supply/demand solution to a feasible circulation.

Example of the reduction

Here is an example:

$$\begin{tikzcd} a^{[4]} \arrow[r, "2"] & b^{[-7]} \arrow[dl, "6"] \arrow[d, "{[0,2]}"] \\ c^{[3]} \arrow[u, "{[1,5]}"] & d^{[0]} \arrow[l, "{[-1,1]}"] \end{tikzcd}$$

We'd solve the resulting supply/demand problem by adding a source s, a sink t, and arcs from s or to t, as we discussed earlier.

Then, we need to convert the resulting supply/demand solution to a feasible circulation.

Example of the reduction

Here is an example:

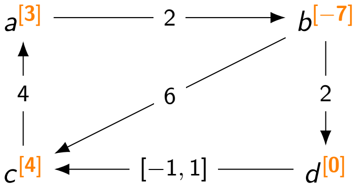
$$\begin{tikzcd}
a^{[4]} \arrow[rr, "2"] & & b^{[-7]} \arrow[dll, "6"] \arrow[d, "2"] \\
c^{[3]} \arrow[u, "{[1,5]}"] & & d^{[0]} \arrow[ll, "{[-1,1]}"]
\end{tikzcd}$$

We'd solve the resulting supply/demand problem by adding a source s, a sink t, and arcs from s or to t, as we discussed earlier.

Then, we need to convert the resulting supply/demand solution to a feasible circulation.

Example of the reduction

Here is an example:

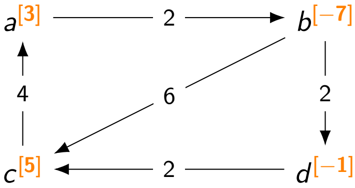


We'd solve the resulting supply/demand problem by adding a source s, a sink t, and arcs from s or to t, as we discussed earlier.

Then, we need to convert the resulting supply/demand solution to a feasible circulation.

Example of the reduction

Here is an example:



We'd solve the resulting supply/demand problem by adding a source s, a sink t, and arcs from s or to t, as we discussed earlier.

Then, we need to convert the resulting supply/demand solution to a feasible circulation.

Converting back to a feasible circulation

If the supply/demand problem is solved by a flow $\mathbf{y}$, the feasible circulation we want is $\mathbf{x}$, where $x_{ij} = y_{ij} + a_{ij}$.

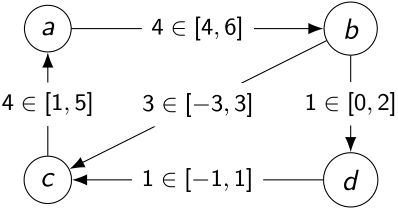
Converting back to a feasible circulation

If the supply/demand problem is solved by a flow $\mathbf{y}$, the feasible circulation we want is $\mathbf{x}$, where $x_{ij} = y_{ij} + a_{ij}$.

$$
\begin{tikzcd}
a^{[3]} \arrow[rr, "0/2"] & & b^{[-7]} \arrow[dll, "6/6"] \arrow[d, "1/2"] \\
c^{[5]} \arrow[u, "3/4"] & & d^{[-1]} \arrow[ll, "2/2"]
\end{tikzcd}
$$

Converting back to a feasible circulation

If the supply/demand problem is solved by a flow **y**, the feasible circulation we want is **x**, where $x_{ij} = y_{ij} + a_{ij}$.

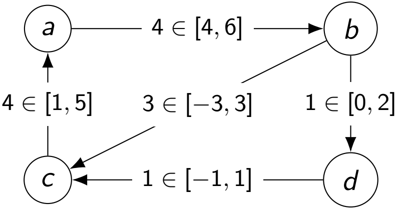


Before the conversion: for each k,

$$\sum_{(i,k)\in A} y_{ik} - \sum_{(k,j)\in A} y_{kj} = d_k$$

Converting back to a feasible circulation

If the supply/demand problem is solved by a flow $\mathbf{y}$, the feasible circulation we want is $\mathbf{x}$, where $x_{ij} = y_{ij} + a_{ij}$.

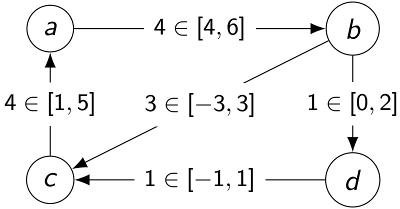


Before the conversion: for each k,

$$\sum_{(i,k)\in A} y_{ik} - \sum_{(k,j)\in A} y_{kj} = d_k = \sum_{j:(k,j)\in A} a_{kj} - \sum_{i:(i,k)\in A} a_{ik}.$$

Converting back to a feasible circulation

If the supply/demand problem is solved by a flow **y**, the feasible circulation we want is **x**, where $x_{ij} = y_{ij} + a_{ij}$.



Before the conversion: for each k,

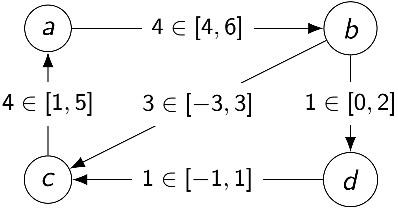
$$\sum_{(i,k)\in A} y_{ik} - \sum_{(k,j)\in A} y_{kj} = d_k = \sum_{j:(k,j)\in A} a_{kj} - \sum_{i:(i,k)\in A} a_{ik}.$$

Therefore

$$\sum_{(i,k)\in A} (y_{ik} + a_{ik}) - \sum_{(k,j)\in A} (y_{kj} + a_{kj}) = 0$$

Converting back to a feasible circulation

If the supply/demand problem is solved by a flow $\mathbf{y}$, the feasible circulation we want is $\mathbf{x}$, where $x_{ij} = y_{ij} + a_{ij}$.



Before the conversion: for each k,

$$\sum_{(i,k)\in A} y_{ik} - \sum_{(k,j)\in A} y_{kj} = d_k = \sum_{j:(k,j)\in A} a_{kj} - \sum_{i:(i,k)\in A} a_{ik}.$$

Therefore

$$\sum_{(i,k)\in A} (y_{ik} + a_{ik}) - \sum_{(k,j)\in A} (y_{kj} + a_{kj}) = 0 \implies \Delta_k(\mathbf{x}) = 0.$$

Bipartite matchings

We have already seen the bipartite matching problem. But we can convert it to a network flow problem, which lets us solve it using Ford–Fulkerson, instead of using linear programming.

Bipartite matchings

We have already seen the bipartite matching problem. But we can convert it to a network flow problem, which lets us solve it using Ford–Fulkerson, instead of using linear programming.

Given a bipartite graph (A, B, E), we:

1. Construct a network with nodes $A \cup B \cup \{s, t\}$.

Bipartite matchings

We have already seen the bipartite matching problem. But we can convert it to a network flow problem, which lets us solve it using Ford–Fulkerson, instead of using linear programming.

Given a bipartite graph (A, B, E), we:

1. Construct a network with nodes $A \cup B \cup \{s, t\}$.
2. For every edge $(i, j) \in E$, add an arc (i, j) with $c_{ij} = \infty$.

Bipartite matchings

We have already seen the bipartite matching problem. But we can convert it to a network flow problem, which lets us solve it using Ford–Fulkerson, instead of using linear programming.

Given a bipartite graph (A, B, E), we:

1. Construct a network with nodes $A \cup B \cup \{s, t\}$.
2. For every edge $(i, j) \in E$, add an arc (i, j) with $c_{ij} = \infty$.
3. For every vertex $i \in A$, add an arc (s, i) with $c_{si} = 1$.
4. For every vertex $j \in B$, add an arc (j, t) with $c_{jt} = 1$.

Bipartite matchings

We have already seen the bipartite matching problem. But we can convert it to a network flow problem, which lets us solve it using Ford–Fulkerson, instead of using linear programming.

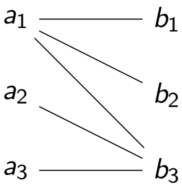
Given a bipartite graph (A, B, E), we:

1. Construct a network with nodes $A \cup B \cup \{s, t\}$.
2. For every edge $(i, j) \in E$, add an arc (i, j) with $c_{ij} = \infty$.
3. For every vertex $i \in A$, add an arc (s, i) with $c_{si} = 1$.
4. For every vertex $j \in B$, add an arc (j, t) with $c_{jt} = 1$.

An maximum flow will send 1 flow along every edge in a matching, and 0 flow along all other edges.

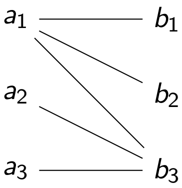
Bipartite matching example

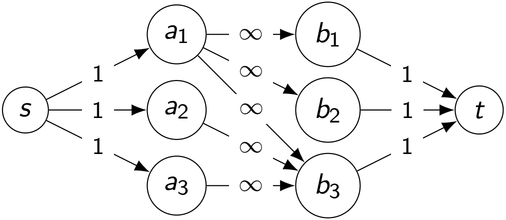
Going from the bipartite graph to a network:



Bipartite matching example

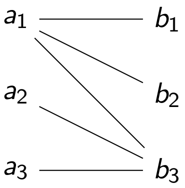
Going from the bipartite graph to a network:

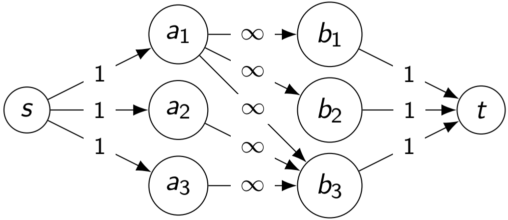




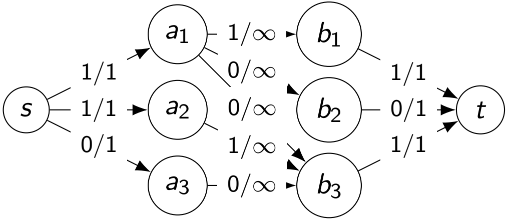
Bipartite matching example

Going from the bipartite graph to a network:



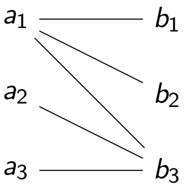


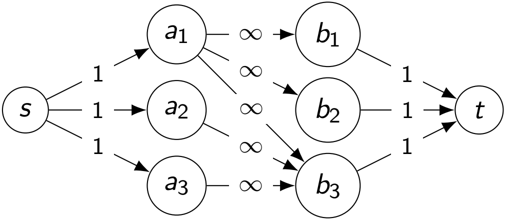
Going from a maximum flow to a maximum matching:



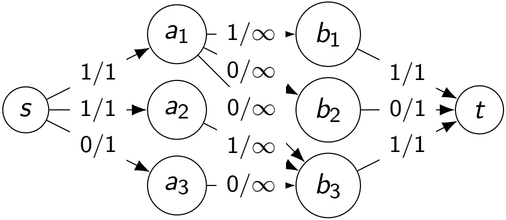
Bipartite matching example

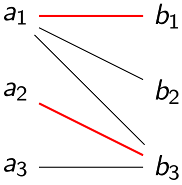
Going from the bipartite graph to a network:





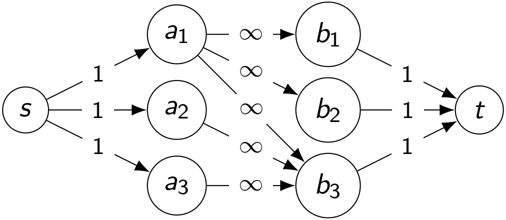
Going from a maximum flow to a maximum matching:





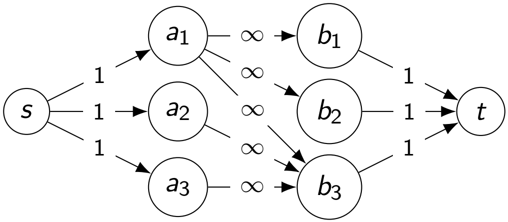
Minimum cuts and vertex covers

What does a cut look like in this picture?



Minimum cuts and vertex covers

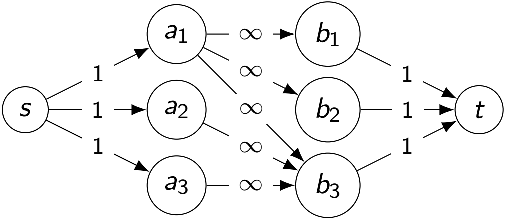
What does a cut look like in this picture?



A cut is *really bad* if one of the ∞ arcs crosses the cut.

Minimum cuts and vertex covers

What does a cut look like in this picture?

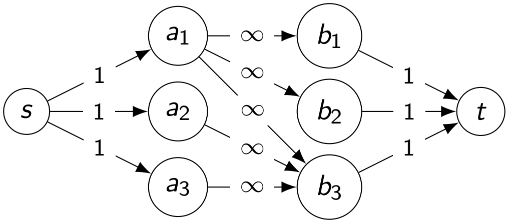


A cut is *really bad* if one of the ∞ arcs crosses the cut.

So there are no arcs from $A \cap S$ to $B \cap T$.

Minimum cuts and vertex covers

What does a cut look like in this picture?



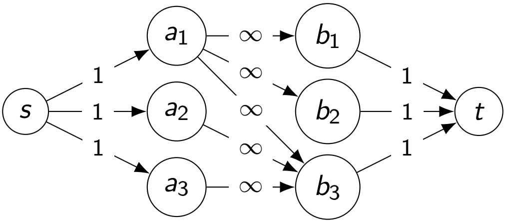
A cut is *really bad* if one of the ∞ arcs crosses the cut.

So there are no arcs from $A \cap S$ to $B \cap T$.

So together, $A \cap T$ and $B \cap S$ form a vertex cover.

Minimum cuts and vertex covers

What does a cut look like in this picture?



A cut is *really bad* if one of the ∞ arcs crosses the cut.

So there are no arcs from $A \cap S$ to $B \cap T$.

So together, $A \cap T$ and $B \cap S$ form a vertex cover. We can check that the capacity of such a cut (S, T) is $|A \cap T| + |B \cap S|$: the number of vertices in the cover.