Minimum-Cost Flow

Math 482, Lecture 28

Misha Lavrov

April 10, 2020

The min-cost flow problem

In this problem, we are given:

- a network (N, A) with no source or sink.

The min-cost flow problem

In this problem, we are given:

- a network (N, A) with no source or sink.
- A demand d_k for every node (as in a supply-demand problem).

The min-cost flow problem

In this problem, we are given:

- a network (N, A) with no source or sink.
- A demand d_k for every node (as in a supply-demand problem).
- Instead of a *capacity* c_{ij} for every arc $(i, j) \in A$, a *cost* c_{ij}.

The min-cost flow problem

In this problem, we are given:

- a network (N, A) with no source or sink.
- A demand d_k for every node (as in a supply-demand problem).
- Instead of a *capacity* c_{ij} for every arc $(i, j) \in A$, a *cost* c_{ij}.

Goal: minimize $\sum_{(i,j)\in A} c_{ij}x_{ij}$ while satisfying $\Delta_k(\mathbf{x}) = d_k$ for every node k (and $\mathbf{x} \geq \mathbf{0}$).

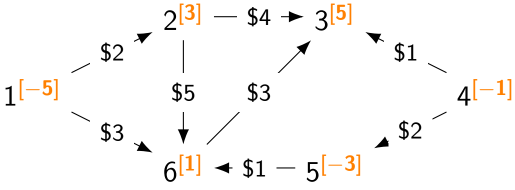
The min-cost flow problem

In this problem, we are given:

- a network (N, A) with no source or sink.
- A demand d_k for every node (as in a supply-demand problem).
- Instead of a *capacity* c_{ij} for every arc $(i, j) \in A$, a *cost* c_{ij}.

Goal: minimize $\sum_{(i,j)\in A} c_{ij}x_{ij}$ while satisfying $\Delta_k(\mathbf{x}) = d_k$ for every node k (and $\mathbf{x} \geq \mathbf{0}$).

Example:



Plan for solving min-cost flow

Unlike max-flow problems, min-cost flow problems are commonly solved using the simplex method. (There's fewer constraints, so the basis is smaller.)

Plan for solving min-cost flow

Unlike max-flow problems, min-cost flow problems are commonly solved using the simplex method. (There's fewer constraints, so the basis is smaller.)

To figure out how to do this, we need to know several things:

- What a basic solution looks like.

Plan for solving min-cost flow

Unlike max-flow problems, min-cost flow problems are commonly solved using the simplex method. (There's fewer constraints, so the basis is smaller.)

To figure out how to do this, we need to know several things:

- What a basic solution looks like.
- How to do a pivoting step.

Plan for solving min-cost flow

Unlike max-flow problems, min-cost flow problems are commonly solved using the simplex method. (There's fewer constraints, so the basis is smaller.)

To figure out how to do this, we need to know several things:

- What a basic solution looks like.
- How to do a pivoting step.
- How to determine the reduced costs of an arc.

Number of basic variables

Our constraints are: $F\mathbf{x} = \mathbf{d}$, where

- F is a $|N| \times |A|$ matrix where x_{ij}'s column has a 1 in row j and a -1 in row i.
- $\mathbf{d}$ is the vector of demands.

Number of basic variables

Our constraints are: $F\mathbf{x} = \mathbf{d}$, where

- F is a $|N| \times |A|$ matrix where x_{ij}'s column has a 1 in row j and a -1 in row i.
- $\mathbf{d}$ is the vector of demands.

Normally, a basic solution would be given by $\mathbf{x} = F_{\mathcal{B}}^{-1}\mathbf{d}$ for some choice of $|N|$ variables $\mathcal{B}$.

Number of basic variables

Our constraints are: $F\mathbf{x} = \mathbf{d}$, where

- F is a $|N| \times |A|$ matrix where x_{ij}'s column has a 1 in row j and a -1 in row i.
- $\mathbf{d}$ is the vector of demands.

Normally, a basic solution would be given by $\mathbf{x} = F_{\mathcal{B}}^{-1}\mathbf{d}$ for some choice of $|N|$ variables $\mathcal{B}$.

Here, one equation is redundant: assuming $\sum_{k \in N} d_k = 0$, the equations add up to $0 = 0$. (If the sum is not 0, there is no solution.)

Number of basic variables

Our constraints are: $F\mathbf{x} = \mathbf{d}$, where

- F is a $|N| \times |A|$ matrix where x_{ij}'s column has a 1 in row j and a -1 in row i.
- $\mathbf{d}$ is the vector of demands.

Normally, a basic solution would be given by $\mathbf{x} = F_{\mathcal{B}}^{-1}\mathbf{d}$ for some choice of $|N|$ variables $\mathcal{B}$.

Here, one equation is redundant: assuming $\sum_{k \in N} d_k = 0$, the equations add up to $0 = 0$. (If the sum is not 0, there is no solution.)

So our basis will have $|N| - 1$ variables, assuming the network is connected.[1]

---

[1] If there are two or more subnetworks with no arcs between them, we solve the subproblems separately.

Spanning trees

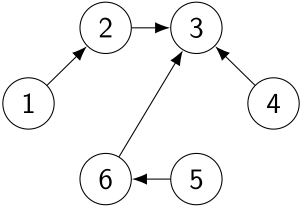
Definition

A *spanning tree* of (N, A) is a choice of $|N| - 1$ arcs forming a connected subnetwork. (For us, connectivity ignores direction.)

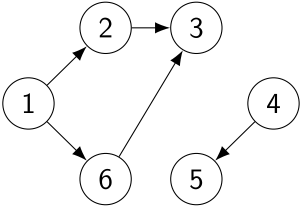
Spanning trees

Definition

A *spanning tree* of (N, A) is a choice of $|N| - 1$ arcs forming a connected subnetwork. (For us, connectivity ignores direction.)



Good

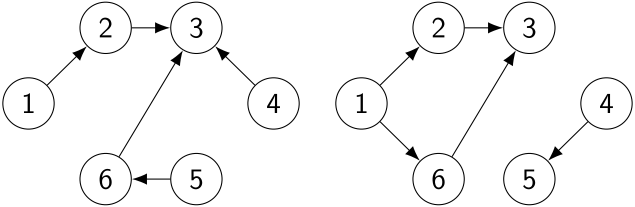


Bad!

Spanning trees

Definition

A *spanning tree* of (N, A) is a choice of $|N| - 1$ arcs forming a connected subnetwork. (For us, connectivity ignores direction.)



Good Bad!

Claim: $|N| - 1$ variables form a basis exactly when their arcs make a spanning tree.

Why spanning trees?

Q1. Why is being a spanning tree necessary to be a basis?

Why spanning trees?

Q1. Why is being a spanning tree necessary to be a basis?

A1. If there are two pieces, then not all systems $F\mathbf{x} = \mathbf{d}$ with $\sum_{k \in N} d_k = 0$ have solutions. We must have the d_k sum to 0 on each piece!

Why spanning trees?

Q1. Why is being a spanning tree necessary to be a basis?

A1. If there are two pieces, then not all systems $F\mathbf{x} = \mathbf{d}$ with $\sum_{k \in N} d_k = 0$ have solutions. We must have the d_k sum to 0 on each piece!

Q2. Why is being a spanning tree sufficient to be a basis?

Why spanning trees?

Q1. Why is being a spanning tree necessary to be a basis?

A1. If there are two pieces, then not all systems $F\mathbf{x} = \mathbf{d}$ with $\sum_{k \in N} d_k = 0$ have solutions. We must have the d_k sum to 0 on each piece!

Q2. Why is being a spanning tree sufficient to be a basis?

A2. We have an algorithm (next slide) to find a basic solution using only arcs in a spanning tree.

Why spanning trees?

Q1. Why is being a spanning tree necessary to be a basis?

A1. If there are two pieces, then not all systems $F\mathbf{x} = \mathbf{d}$ with $\sum_{k \in N} d_k = 0$ have solutions. We must have the d_k sum to 0 on each piece!

Q2. Why is being a spanning tree sufficient to be a basis?

A2. We have an algorithm (next slide) to find a basic solution using only arcs in a spanning tree.

This will give us an $\mathbf{x}$ such that $F\mathbf{x} = \mathbf{d}$, but not necessarily $\mathbf{x} \geq \mathbf{0}$.

From a spanning tree to a basic solution

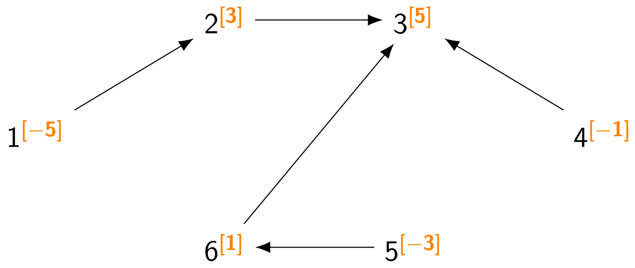
To find a basic solution, repeat the following:

1. Pick a node k with only one arc of the spanning tree with unknown flow in/out of k.
2. Solve for that remaining flow to make $\Delta_k(\mathbf{x}) = d_k$.

From a spanning tree to a basic solution

To find a basic solution, repeat the following:

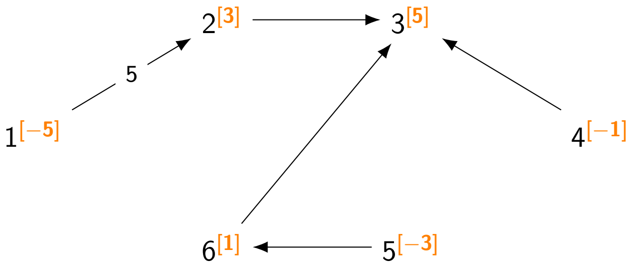
1. Pick a node k with only one arc of the spanning tree with unknown flow in/out of k.
2. Solve for that remaining flow to make $\Delta_k(\mathbf{x}) = d_k$.



From a spanning tree to a basic solution

To find a basic solution, repeat the following:

1. Pick a node k with only one arc of the spanning tree with unknown flow in/out of k.
2. Solve for that remaining flow to make $\Delta_k(\mathbf{x}) = d_k$.

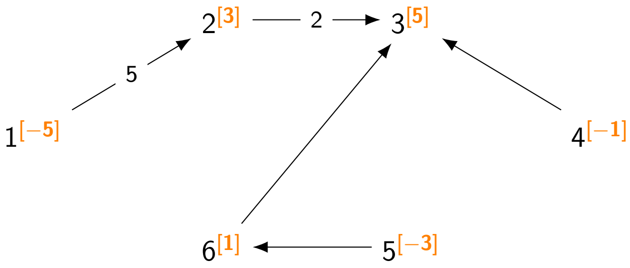


Solve for x_{12} using node 1

From a spanning tree to a basic solution

To find a basic solution, repeat the following:

1. Pick a node k with only one arc of the spanning tree with unknown flow in/out of k.
2. Solve for that remaining flow to make $\Delta_k(\mathbf{x}) = d_k$.

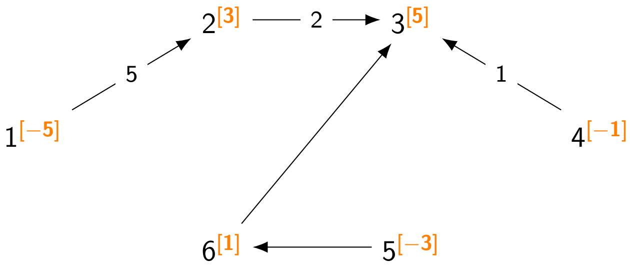


Solve for x_{23} using node 2

From a spanning tree to a basic solution

To find a basic solution, repeat the following:

1. Pick a node k with only one arc of the spanning tree with unknown flow in/out of k.
2. Solve for that remaining flow to make $\Delta_k(\mathbf{x}) = d_k$.

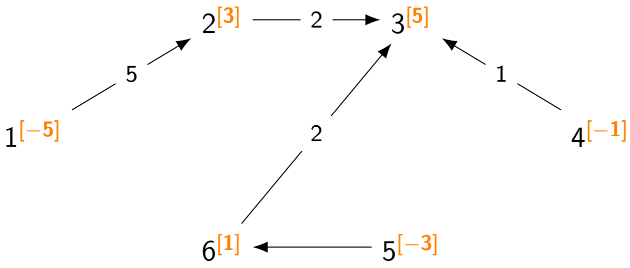


Solve for x_{43} using node 4

From a spanning tree to a basic solution

To find a basic solution, repeat the following:

1. Pick a node k with only one arc of the spanning tree with unknown flow in/out of k.
2. Solve for that remaining flow to make $\Delta_k(\mathbf{x}) = d_k$.

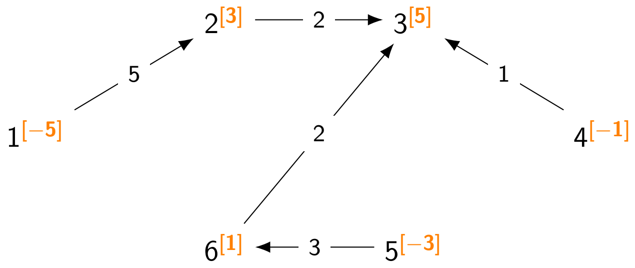


Solve for x_{63} using node 3

From a spanning tree to a basic solution

To find a basic solution, repeat the following:

1. Pick a node k with only one arc of the spanning tree with unknown flow in/out of k.
2. Solve for that remaining flow to make $\Delta_k(\mathbf{x}) = d_k$.



Solve for x_{56} using node 5

Ideas about pivoting steps

Things we want from pivoting:

- We should be able to add any arc we want, and remove an existing arc to get another spanning tree.

Ideas about pivoting steps

Things we want from pivoting:

- We should be able to add any arc we want, and remove an existing arc to get another spanning tree.
- If we have a basic feasible solution $(F\mathbf{x} = \mathbf{d},\ \mathbf{x} \geq \mathbf{0})$ before the pivot, that should stay true after the pivot.

Ideas about pivoting steps

Things we want from pivoting:

- We should be able to add any arc we want, and remove an existing arc to get another spanning tree.
- If we have a basic feasible solution ($F\mathbf{x} = \mathbf{d}$, $\mathbf{x} \geq \mathbf{0}$) before the pivot, that should stay true after the pivot.

Vague idea: adding an arc to a spanning tree creates a cycle!

We can modify the flows along that cycle to get infinitely many new solutions.

Ideas about pivoting steps

Things we want from pivoting:

- We should be able to add any arc we want, and remove an existing arc to get another spanning tree.
- If we have a basic feasible solution ($F\mathbf{x} = \mathbf{d}$, $\mathbf{x} \geq \mathbf{0}$) before the pivot, that should stay true after the pivot.

Vague idea: adding an arc to a spanning tree creates a cycle!

We can modify the flows along that cycle to get infinitely many new solutions.

One of those solutions will have flow $x_{ij} = 0$ for an existing arc (i, j).

The pivoting step algorithm

Once we've picked an arc (i, j) we're adding to the basis:

1. Find the cycle formed by that arc and the spanning tree.

The pivoting step algorithm

Once we've picked an arc (i, j) we're adding to the basis:

1. Find the cycle formed by that arc and the spanning tree.
2. Set $x_{ij} = \delta$. For each arc on the cycle, **add** δ if it has the same direction around the cycle as (i, j), **subtract** δ otherwise.

The pivoting step algorithm

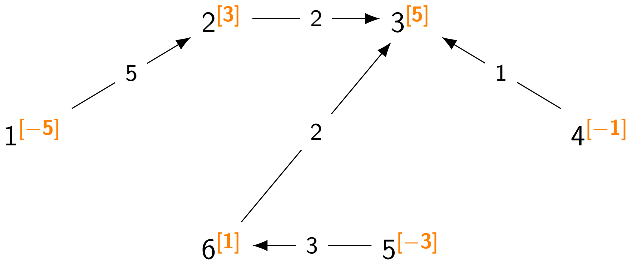
Once we've picked an arc (i, j) we're adding to the basis:

1. Find the cycle formed by that arc and the spanning tree.
2. Set $x_{ij} = \delta$. For each arc on the cycle, **add** δ if it has the same direction around the cycle as (i, j), **subtract** δ otherwise.
3. Find the smallest δ that sets an existing arc to 0.

The pivoting step algorithm

Once we've picked an arc (i, j) we're adding to the basis:

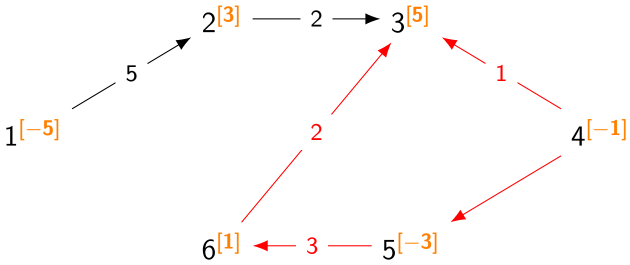
1. Find the cycle formed by that arc and the spanning tree.
2. Set $x_{ij} = \delta$. For each arc on the cycle, **add** δ if it has the same direction around the cycle as (i, j), **subtract** δ otherwise.
3. Find the smallest δ that sets an existing arc to 0.



The pivoting step algorithm

Once we've picked an arc (i, j) we're adding to the basis:

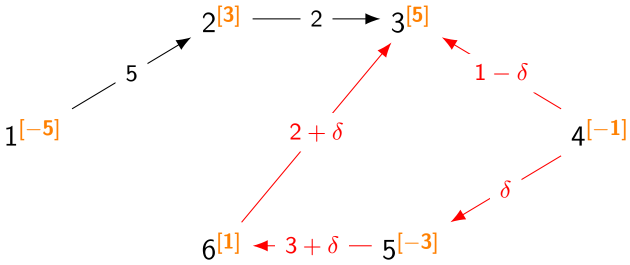
1. Find the cycle formed by that arc and the spanning tree.
2. Set $x_{ij} = \delta$. For each arc on the cycle, **add** δ if it has the same direction around the cycle as (i, j), **subtract** δ otherwise.
3. Find the smallest δ that sets an existing arc to 0.



The pivoting step algorithm

Once we've picked an arc (i,j) we're adding to the basis:

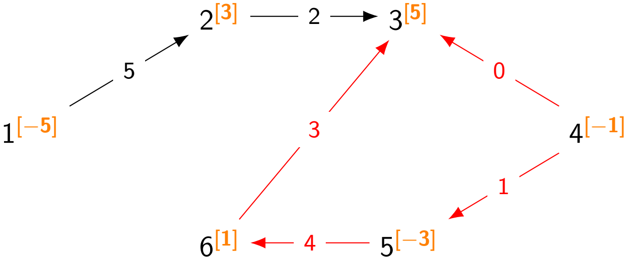
1. Find the cycle formed by that arc and the spanning tree.
2. Set $x_{ij} = \delta$. For each arc on the cycle, **add** δ if it has the same direction around the cycle as (i,j), **subtract** δ otherwise.
3. Find the smallest δ that sets an existing arc to 0.



The pivoting step algorithm

Once we've picked an arc (i, j) we're adding to the basis:

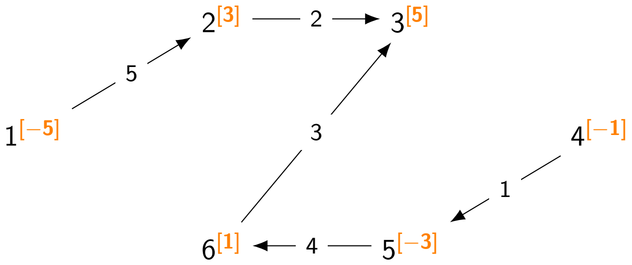
1. Find the cycle formed by that arc and the spanning tree.
2. Set $x_{ij} = \delta$. For each arc on the cycle, **add** δ if it has the same direction around the cycle as (i, j), **subtract** δ otherwise.
3. Find the smallest δ that sets an existing arc to 0.



The pivoting step algorithm

Once we've picked an arc (i, j) we're adding to the basis:

1. Find the cycle formed by that arc and the spanning tree.
2. Set $x_{ij} = \delta$. For each arc on the cycle, **add** δ if it has the same direction around the cycle as (i, j), **subtract** δ otherwise.
3. Find the smallest δ that sets an existing arc to 0.



Reduced costs

Q. How can we compute the reduced cost of an arc we're adding?

Reduced costs

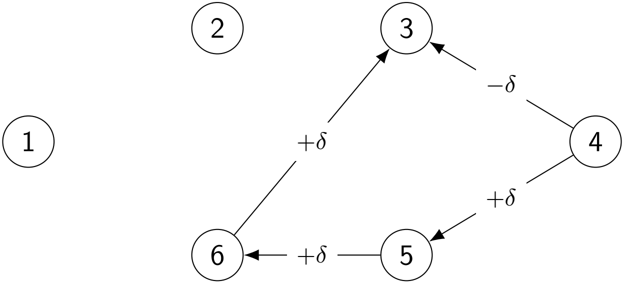
Q. How can we compute the reduced cost of an arc we're adding?

A. Look at the $\pm\delta$ changes and multiply them by the costs of the arcs.

Reduced costs

Q. How can we compute the reduced cost of an arc we're adding?

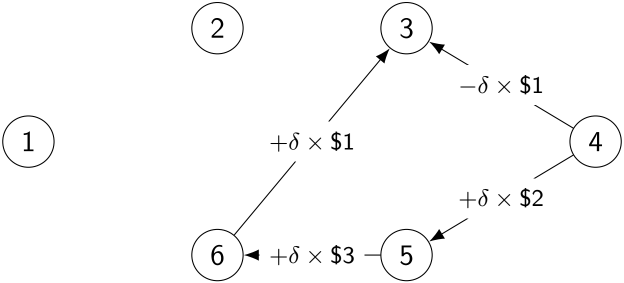
A. Look at the $\pm\delta$ changes and multiply them by the costs of the arcs.



Reduced costs

Q. How can we compute the reduced cost of an arc we're adding?

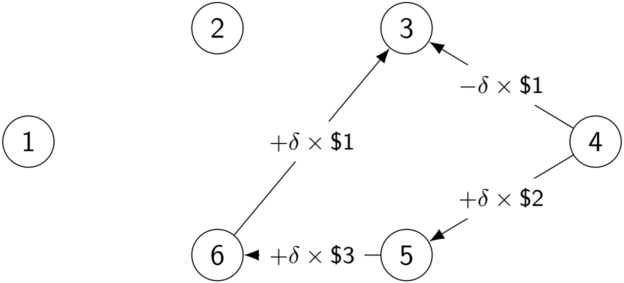
A. Look at the $\pm\delta$ changes and multiply them by the costs of the arcs.



Reduced costs

Q. How can we compute the reduced cost of an arc we're adding?

A. Look at the $\pm\delta$ changes and multiply them by the costs of the arcs.



Total change in cost: $\delta \times (2 + 3 + 1 - 1) = 5\delta$, so the reduced cost of x_{45} is 5.

A two-phase method for min-cost flow

We now know how to find the min-cost flow once we have a starting basic feasible flow. But how can we get that starting point?

A two-phase method for min-cost flow

We now know how to find the min-cost flow once we have a starting basic feasible flow. But how can we get that starting point?

The two-phase simplex method:

1. Add artificial variables to each constraint, so that we get a basic feasible solution using only artificial variables.
2. Add an artificial objective function that tries to force out those variables.

A two-phase method for min-cost flow

We now know how to find the min-cost flow once we have a starting basic feasible flow. But how can we get that starting point?

The two-phase simplex method:

1. Add artificial variables to each constraint, so that we get a basic feasible solution using only artificial variables.

   Add an artificial node a. For each node k: if $d_k > 0$, add arc (a, k) with $x_{ak} = d_k$; if $d_k < 0$, add arc (k, a) with $x_{ka} = |d_k|$.

2. Add an artificial objective function that tries to force out those variables.

A two-phase method for min-cost flow

We now know how to find the min-cost flow once we have a starting basic feasible flow. But how can we get that starting point?

The two-phase simplex method:

1. Add artificial variables to each constraint, so that we get a basic feasible solution using only artificial variables.

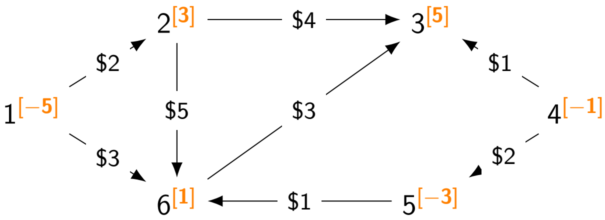
   Add an artificial node a. For each node k: if $d_k > 0$, add arc (a, k) with $x_{ak} = d_k$; if $d_k < 0$, add arc (k, a) with $x_{ka} = |d_k|$.

2. Add an artificial objective function that tries to force out those variables.

   Set the cost to \$1 for artificial arcs, \$0 for original arcs.

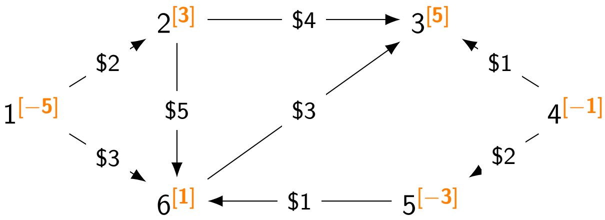
Example of the first phase

Original min-cost flow problem:

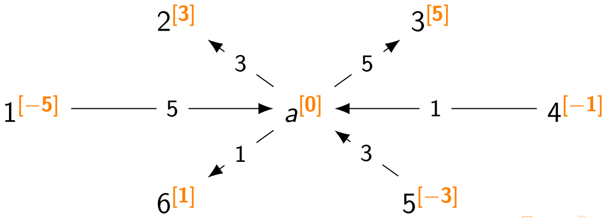


Example of the first phase

Original min-cost flow problem:



Initial spanning tree for the phase-one problem:



Moving from first to second phase

In the usual simplex method, it's typical that we'll be able to drive out all artificial variables.

Moving from first to second phase

In the usual simplex method, it's typical that we'll be able to drive out all artificial variables.

Here:

- Any spanning tree with the artificial node a in it must include some arc in or out of a, to be connected.

Moving from first to second phase

In the usual simplex method, it's typical that we'll be able to drive out all artificial variables.

Here:

- Any spanning tree with the artificial node a in it must include some arc in or out of a, to be connected.
- We stop when there is only one arc in or out of a left in the spanning tree.

Moving from first to second phase

In the usual simplex method, it's typical that we'll be able to drive out all artificial variables.

Here:

- Any spanning tree with the artificial node a in it must include some arc in or out of a, to be connected.
- We stop when there is only one arc in or out of a left in the spanning tree.
- Because we want $\Delta_a(\mathbf{x}) = d_a = 0$, that arc must have flow 0.

Moving from first to second phase

In the usual simplex method, it's typical that we'll be able to drive out all artificial variables.

Here:

- Any spanning tree with the artificial node a in it must include some arc in or out of a, to be connected.
- We stop when there is only one arc in or out of a left in the spanning tree.
- Because we want $\Delta_a(\mathbf{x}) = d_a = 0$, that arc must have flow 0.

Once we delete the artificial node and artificial arcs, we're left with a basic feasible solution to the original problem.