



SAPIENZA
UNIVERSITÀ DI ROMA

Network Optimization

Volume 3 — Flows

Network flow models and total unimodularity; maximum flow and minimum cut; minimum cost flow with negative cycles, potentials and reduced costs; transportation, transshipment and assignment problems.

Fabio Furini

fabiofurini.github.io

Department of Computer, Control and Management Engineering — Sapienza University of Rome

Online version and downloadable material:
fabiofurini.github.io/network-optimization/
github.com/fabiofurini/network-optimization

Academic year 2026–2027

Contents

4	Network flow models	3
4.1	Networks and flows	4
4.2	The minimum cost flow model	6
4.3	Feasibility conditions	7
4.4	Removing the lower bounds	8
4.5	Unbalanced instances	9
4.6	Total unimodularity and integrality	11
4.7	The dual model and the reduced costs	13
4.8	Special cases	16
4.9	A transshipment instance	16
4.10	Typical mistakes	18
4.11	In summary	18
5	Maximum flow	20
5.1	The maximum flow problem	21
5.2	Cuts	23
5.3	The residual network	25
5.4	The max-flow min-cut theorem	27
5.5	The Ford–Fulkerson algorithm	29
5.6	The Edmonds–Karp algorithm	32
5.7	Applications	35
5.8	Typical mistakes	38
5.9	In summary	38
6	Minimum cost flow	40
6.1	The residual network with costs	41
6.2	Circulations and their decomposition	43
6.3	Negative cycle optimality conditions	45
6.4	Optimality conditions with potentials	47
6.5	A first feasible flow	48
6.6	The cycle-canceling algorithm	50
6.7	The successive shortest path algorithm	54
6.8	Typical mistakes	59



6.9	In summary	59
7	Transportation and assignment	61
7.1	The transportation problem	62
7.2	Basic solutions	66
7.3	Initial basic feasible solutions	69
7.4	The potential method	72
7.5	The assignment problem	80
7.6	Typical mistakes	87
7.7	In summary	87

Network flow models

Prerequisites

- From this module, [Chapter 1 «Graphs»](#): directed graphs, the node–arc incidence matrix, spanning trees and the rank of the incidence matrix.
- From the module Linear Programming, [Chapter 9 «Duality theory»](#) and [Chapter 15 «The geometry of Linear Programming»](#): the dual of an LP, weak and strong duality, vertices of a polyhedron.
- Determinants: the Laplace expansion along a column and Cramer’s rule.

Objectives

At the end of this chapter you will be able to:

- describe a network with balances, lower bounds, capacities and costs, and write the minimum cost flow model, also in matrix form with the incidence matrix;
- check the necessary conditions for the existence of a feasible flow, and build counterexamples;
- remove the lower bounds by a substitution, and balance an instance with a dummy node;
- prove that the incidence matrix is totally unimodular, and deduce that the model has integer optimal solutions when the data are integer;
- write the dual with the potentials π_i , compute the reduced costs and certify the optimality of a flow;
- recognize the shortest path, the maximum flow, the transportation and the assignment problems as minimum cost flow models.

4.1 Networks and flows

Definition 4.1: network

A **network** is a directed graph $G = (N, A)$, with $n = |N|$ nodes and $m = |A|$ arcs, together with the following data.

- Every node $i \in N$ has a **balance** $b_i \in \mathbb{R}$. The node is a **supply node** if $b_i > 0$, a **demand node** if $b_i < 0$, and a **transshipment node** if $b_i = 0$.
- Every arc $(i, j) \in A$ has a **lower bound** l_{ij} and a **capacity** u_{ij} , with $0 \leq l_{ij} \leq u_{ij} \leq +\infty$, and a **cost** $c_{ij} \in \mathbb{R}$ per unit of flow.

When the lower bounds are not given, $l_{ij} = 0$ for every arc.

Definition 4.2: flow

Let $G = (N, A)$ be a network. A **flow** is a vector $\mathbf{x} \in \mathbb{R}^m$ with one entry x_{ij} , the amount sent along the arc, for every $(i, j) \in A$.

- The flow satisfies the **flow conservation** constraint of node i if what leaves i minus what enters i equals the balance:

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} - \sum_{(j,i) \in \delta^-(i)} x_{ji} = b_i.$$

- The flow is **feasible** if it satisfies flow conservation at every node and $l_{ij} \leq x_{ij} \leq u_{ij}$ for every arc. Its **cost** is $\sum_{(i,j) \in A} c_{ij} x_{ij}$.

- A supply node sends out b_i units more than it receives, a demand node receives $-b_i$ units more than it sends, and a transshipment node passes on everything it receives.
- With the node-arc incidence matrix $\mathbf{B}$ of Chapter 1 «Graphs», flow conservation at all the nodes reads $\mathbf{B}\mathbf{x} = \mathbf{b}$: row i of $\mathbf{B}\mathbf{x}$ is the outflow minus the inflow of i .

Example 4.1: the reference network

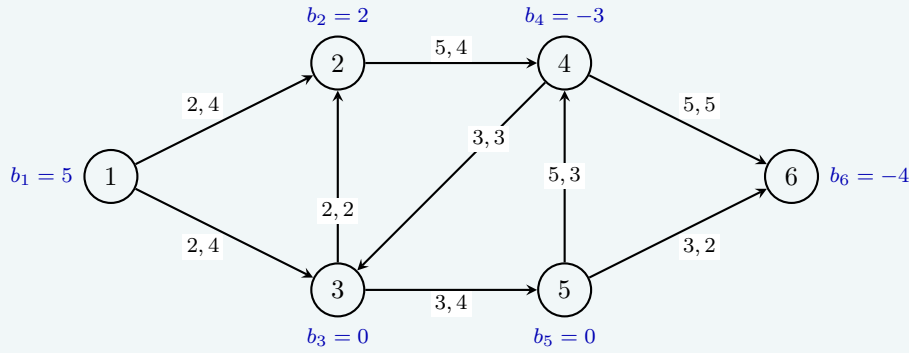
We add data to the reference graph of Chapter 1 «Graphs», with the arcs numbered $a_1, a_2, \dots, a_9$ in lexicographic order. The balances are

$$b_1 = 5, \quad b_2 = 2, \quad b_3 = 0, \quad b_4 = -3, \quad b_5 = 0, \quad b_6 = -4,$$

so nodes 1 and 2 supply 7 units, nodes 4 and 6 demand them, and nodes 3 and 5 are transshipment nodes. The lower bounds are zero, and the costs and capacities are:

arc	(1, 2)	(1, 3)	(2, 4)	(3, 2)	(3, 5)	(4, 3)	(4, 6)	(5, 4)	(5, 6)
c_{ij}	2	2	5	2	3	3	5	5	3
u_{ij}	4	4	4	2	4	3	5	3	2

In the figure each arc carries the pair c_{ij}, u_{ij} , and each node its balance b_i (in blue).



The flow $\bar{x}$ with

$$\bar{x}_{12} = 1, \bar{x}_{13} = 4, \bar{x}_{24} = 3, \bar{x}_{32} = 0, \bar{x}_{35} = 4, \bar{x}_{43} = 0, \bar{x}_{46} = 2, \bar{x}_{54} = 2, \bar{x}_{56} = 2$$

is feasible. For example, at node 4 the outflow is $\bar{x}_{43} + \bar{x}_{46} = 2$, the inflow is $\bar{x}_{24} + \bar{x}_{54} = 5$, and $2 - 5 = -3 = b_4$. The arcs $(1, 3)$, $(3, 5)$ and $(5, 6)$ are at capacity, and the cost of $\bar{x}$ is 63.

4.1.1 Flow across a cut

- For a set of nodes $S \subseteq N$, we denote by $\delta^+(S)$ the arcs with the tail in S and the head outside S , by $\delta^-(S)$ the arcs with the head in S and the tail outside S , and $x(F) = \sum_{(i,j) \in F} x_{ij}$ for a set of arcs F .

Lemma 4.3: flow across a cut

Let x satisfy flow conservation at every node of a network $G = (N, A)$. Then, for every $S \subseteq N$,

$$x(\delta^+(S)) - x(\delta^-(S)) = \sum_{i \in S} b_i.$$

Proof. Add up the conservation constraints of the nodes of S ; the right-hand side is $\sum_{i \in S} b_i$.

- An arc with both endpoints in S appears twice, with $+x_{ij}$ in the row of its tail and $-x_{ij}$ in the row of its head: it cancels.
- An arc of $\delta^+(S)$ appears only in the row of its tail, with $+x_{ij}$; an arc of $\delta^-(S)$ only in the row of its head, with $-x_{ij}$; the other arcs do not appear. $\square$

- In [Example 4.1](#), the set $S = \{1, 2, 3\}$ has $\delta^+(S) = \{(2, 4), (3, 5)\}$ and $\delta^-(S) = \{(4, 3)\}$. The flow $\bar{x}$ gives $3 + 4 - 0 = 7 = b_1 + b_2 + b_3$.

A network has balances on the nodes and lower bounds, capacities and costs on the arcs. A feasible flow satisfies $Bx = b$ and the bounds; across every cut, the net outflow equals the total balance of the nodes inside.

4.2 The minimum cost flow model

Definition 4.4: minimum cost flow problem

Given a network $G = (N, A)$ with balances b_i , lower bounds l_{ij} , capacities u_{ij} and costs c_{ij} , the **minimum cost flow problem** asks for a feasible flow of minimum cost.

- We use one continuous variable x_{ij} for every arc $(i, j) \in A$, the flow on the arc.

Using these variables, a Linear Programming (LP) model for the minimum cost flow problem reads as follows:

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij} \quad (4.1a)$$

$$\text{subject to } \sum_{(i,j) \in \delta^+(i)} x_{ij} - \sum_{(j,i) \in \delta^-(i)} x_{ji} = b_i, \quad \forall i \in N, \quad (4.1b)$$

$$l_{ij} \leq x_{ij} \leq u_{ij}, \quad \forall (i, j) \in A. \quad (4.1c)$$

This LP model is referred to as the MCF model.

- Description of the objective function and constraints of the MCF model:
 - the linear objective function (4.1a) minimizes the total cost of the flow;
 - constraints (4.1b) are the flow conservation constraints, one for every node;
 - finally, constraints (4.1c) impose the lower bound and the capacity of every arc.
- With the node–arc incidence matrix $\mathbf{B}$, the vector $\mathbf{c}$ of the costs and the vectors $\mathbf{l}$, $\mathbf{u}$ of the bounds, the MCF model is

$$\min \{ \mathbf{c}'\mathbf{x} : \mathbf{B}\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \}.$$

- The model has m variables and n equations. The column of x_{ij} in $\mathbf{B}$ has only two nonzero entries, +1 in row i and -1 in row j : this structure gives all the properties of this chapter.

Example 4.2: the MCF model of the reference network

For the network of [Example 4.1](#) the MCF model has 9 variables and 6 equations:

$$\begin{array}{lllllllll} \min & 2x_{12} & +2x_{13} & +5x_{24} & +2x_{32} & +3x_{35} & +3x_{43} & +5x_{46} & +5x_{54} & +3x_{56} \\ \text{subject to} & x_{12} & +x_{13} & & & & & & & & = 5 \\ & -x_{12} & & +x_{24} & -x_{32} & & & & & & = 2 \\ & & -x_{13} & & +x_{32} & +x_{35} & -x_{43} & & & & = 0 \\ & & & -x_{24} & & & +x_{43} & +x_{46} & -x_{54} & & = -3 \\ & & & & & -x_{35} & & & +x_{54} & +x_{56} & = 0 \\ & & & & & & & -x_{46} & & -x_{56} & = -4 \end{array}$$

together with $0 \leq x_{12} \leq 4$, $0 \leq x_{13} \leq 4$, $0 \leq x_{24} \leq 4$, $0 \leq x_{32} \leq 2$, $0 \leq x_{35} \leq 4$,

$0 \leq x_{43} \leq 3, 0 \leq x_{46} \leq 5, 0 \leq x_{54} \leq 3, 0 \leq x_{56} \leq 2$. The coefficient matrix of the equations is the incidence matrix $\mathbf{B}$ of the reference graph.

The minimum cost flow problem is the LP $\min\{\mathbf{c}'\mathbf{x} : \mathbf{B}\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$: one variable per arc, one conservation equation per node.

4.3 Feasibility conditions

Definition 4.5: balanced network

A network is **balanced** if $\sum_{i \in N} b_i = 0$, that is, if the total supply equals the total demand.

Proposition 4.6: a feasible network is balanced

If a network $G = (N, A)$ has a feasible flow, then $\sum_{i \in N} b_i = 0$.

Proof. Apply Lemma 4.3 with $S = N$: no arc leaves or enters N , so $\delta^+(N) = \delta^-(N) = \emptyset$ and $\sum_{i \in N} b_i = 0$. In matrix form, the rows of $\mathbf{B}$ add up to the zero vector, hence so do the entries of $\mathbf{b} = \mathbf{B}\mathbf{x}$. $\square$

Proposition 4.7: cut condition

If a network $G = (N, A)$ has a feasible flow, then for every $S \subseteq N$

$$\sum_{i \in S} b_i \leq u(\delta^+(S)) - l(\delta^-(S)),$$

where $u(F)$ and $l(F)$ are the sums of the capacities and of the lower bounds of the arcs of F .

Proof. Let $\mathbf{x}$ be a feasible flow. By Lemma 4.3 and the bounds $x_{ij} \leq u_{ij}$ on $\delta^+(S)$ and $x_{ij} \geq l_{ij}$ on $\delta^-(S)$,

$$\sum_{i \in S} b_i = x(\delta^+(S)) - x(\delta^-(S)) \leq u(\delta^+(S)) - l(\delta^-(S)). \quad \square$$

- The supply of the nodes of S , net of their demand, must be able to leave S through the arcs of $\delta^+(S)$, while the arcs of $\delta^-(S)$ bring at least their lower bounds into S .
- A balanced network can still have no feasible flow: the balance is only a necessary condition.

Example 4.3: balanced networks without a feasible flow

We keep the arcs, costs and capacities of the reference network and change the balances.

1. With $b_5 = -1$, $b_6 = 1$ and $b_i = 0$ for the other nodes, the network is balanced. But node 6 has no leaving arc: for $S = \{6\}$ we get $\delta^+(S) = \emptyset$, and the cut condition fails, since $b_6 = 1 > 0$.
2. With $b_1 = 9$, $b_6 = -8$ and the other balances as in [Example 4.1](#), the network is balanced, since $9+2-3-8 = 0$. For $S = \{1\}$ we get $u(\delta^+(S)) = u_{12}+u_{13} = 8 < 9 = b_1$: node 1 cannot send out its supply.

For the reference network itself, all the 62 sets S with $\emptyset \neq S \neq N$ satisfy the cut condition, and the flow $\bar{x}$ of [Example 4.1](#) is feasible.

A network with a feasible flow is balanced, and every cut can carry the net supply of the nodes inside it. These conditions are necessary: a balanced network may have no feasible flow.

4.4 Removing the lower bounds

Definition 4.8: substitution of the lower bounds

Given a network with lower bounds l , the **substituted network** has the same nodes, arcs and costs, zero lower bounds and

$$b'_i = b_i - \sum_{(i,j) \in \delta^+(i)} l_{ij} + \sum_{(j,i) \in \delta^-(i)} l_{ji}, \quad \forall i \in N, \quad u'_{ij} = u_{ij} - l_{ij}, \quad \forall (i,j) \in A,$$

that is, $b' = b - Bl$ and $u' = u - l$.

- The l_{ij} units that must travel on (i,j) are sent in advance: the tail i has l_{ij} units less to send, the head j has l_{ij} units less to receive, and the arc has l_{ij} units less of free capacity.

Proposition 4.9: removing the lower bounds

The map $x' = x - l$ is a one-to-one correspondence between the feasible flows of a network and those of its substituted network, and

$$c'x = c'x' + c'l.$$

Hence the two MCF models have the same optimal solutions, up to the shift by l , and their optimal values differ by the constant $c'l$.

Proof. The map is linear and invertible, with inverse $x = x' + l$. For every $x \in \mathbb{R}^m$:

1. $Bx = b$ if and only if $Bx' = Bx - Bl = b - Bl = b'$;
2. $l \leq x \leq u$ if and only if $0 \leq x' \leq u - l = u'$;

$$3. \mathbf{c}'\mathbf{x} = \mathbf{c}'(\mathbf{x}' + \mathbf{l}) = \mathbf{c}'\mathbf{x}' + \mathbf{c}'\mathbf{l}.$$

The cost of every feasible flow changes by the same constant, so the optimal solutions correspond. $\square$

Example 4.4: a lower bound on the reference network

In the network of [Example 4.1](#) we impose $l_{32} = 1$: at least one unit must travel from 3 to 2. The column of $(3, 2)$ in $\mathbf{B}$ has $+1$ in row 3 and -1 in row 2, so $\mathbf{B}\mathbf{l} = (0, -1, 1, 0, 0, 0)'$ and

$$\mathbf{b}' = \mathbf{b} - \mathbf{B}\mathbf{l} = (5, 3, -1, -3, 0, -4)', \quad u'_{32} = 2 - 1 = 1, \quad \mathbf{c}'\mathbf{l} = c_{32} l_{32} = 2.$$

Node 3 becomes a demand node, because it has already sent one unit, and node 2 has one more unit to send. The substituted network has optimal value 60; adding the constant 2, the optimal value with $l_{32} = 1$ is 62.

The substitution $\mathbf{x} = \mathbf{l} + \mathbf{x}'$ removes the lower bounds: $\mathbf{b}' = \mathbf{b} - \mathbf{B}\mathbf{l}$, $\mathbf{u}' = \mathbf{u} - \mathbf{l}$, and the objective changes by the constant $\mathbf{c}'\mathbf{l}$.

4.5 Unbalanced instances

Definition 4.10: unbalanced instance

An instance with $\sum_{i \in N} b_i > 0$ is **unbalanced**: the total supply exceeds the total demand. Every demand must be met, and every supply node i sends out *at most* b_i units:

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} - \sum_{(j,i) \in \delta^-(i)} x_{ji} \leq b_i \quad \forall i \in N : b_i > 0, \quad = b_i \quad \forall i \in N : b_i \leq 0,$$

with the bounds $l_{ij} \leq x_{ij} \leq u_{ij}$ and the objective $\min \sum_{(i,j) \in A} c_{ij} x_{ij}$.

Definition 4.11: dummy node

Let $P = \{i \in N : b_i > 0\}$ be the supply nodes and $e = \sum_{i \in N} b_i > 0$ the excess. The **balanced network** adds a **dummy node** d with balance $b_d = -e$, and an arc (i, d) for every $i \in P$ with $l_{id} = 0$, $u_{id} = +\infty$ and $c_{id} = 0$.

Proposition 4.12: the dummy node

The feasible solutions of the unbalanced instance and the feasible flows of the balanced network correspond, with the same cost: a flow of the balanced network restricted to A is feasible for the unbalanced instance, and a solution of the unbalanced instance becomes a feasible flow with $x_{id} = b_i - (\text{outflow} - \text{inflow of } i)$.

Proof. For a vector $\mathbf{x}$ on A , let $s_i(\mathbf{x})$ be the outflow minus the inflow of node i along the arcs of A .

1. Let $(\mathbf{x}, (x_{id})_{i \in P})$ be feasible for the balanced network. At $i \in P$ conservation reads $s_i(\mathbf{x}) + x_{id} = b_i$ with $x_{id} \geq 0$, so $s_i(\mathbf{x}) \leq b_i$; at the other nodes of N it reads $s_i(\mathbf{x}) = b_i$.
2. Conversely, let $\mathbf{x}$ be feasible for the unbalanced instance, and set $x_{id} = b_i - s_i(\mathbf{x}) \geq 0$ for $i \in P$. Conservation holds at every node of N , and at d the inflow is

$$\sum_{i \in P} x_{id} = \sum_{i \in P} b_i - \sum_{i \in N} s_i(\mathbf{x}) + \sum_{i \notin P} s_i(\mathbf{x}) = \sum_{i \in P} b_i - 0 + \sum_{i \notin P} b_i = e,$$

since the values $s_i(\mathbf{x})$ add up to zero: every arc of A contributes $+x_{ij}$ to its tail and $-x_{ij}$ to its head. Hence d receives exactly $-b_d = e$ units.

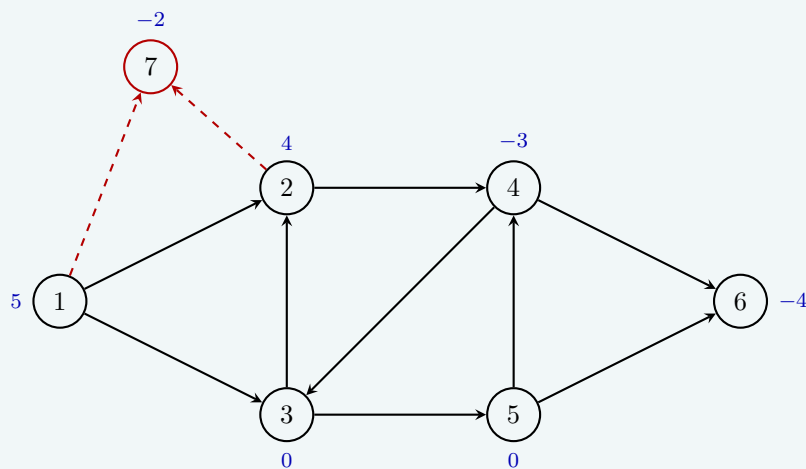
The dummy arcs cost zero, so the two costs coincide. $\square$

- The flow on (i, d) is the part of the supply of i that is not used.
- If the total demand exceeds the total supply and the demands are upper limits, a dummy node d with balance $b_d = -\sum_{i \in N} b_i$ and arcs (d, j) to the demand nodes balances the network in the same way.

Example 4.5: a dummy node on the reference network

In the network of [Example 4.1](#) we raise the supply of node 2 to $b_2 = 4$. The total supply is 9, the total demand is 7, and the excess is $e = 2$.

- The balanced network adds node 7 with $b_7 = -2$, and the arcs $(1, 7)$ and $(2, 7)$ with cost 0 and capacity $+\infty$ (dashed in the figure).
- Its optimal flow, computed with Gurobi, ships all the 4 units of node 2 and only 3 units of node 1: $x_{17} = 2$, $x_{27} = 0$, with optimal value 56.



An unbalanced instance becomes a balanced network with a dummy node that absorbs the excess through zero-cost arcs; the flow on a dummy arc is the unused supply of its tail.

4.6 Total unimodularity and integrality

Definition 4.13: totally unimodular matrix

An integer matrix $\mathbf{M}$ is **totally unimodular** (TU) if the determinant of every square submatrix of $\mathbf{M}$ is 0, +1 or -1. A square submatrix is obtained by choosing k rows and k columns, for any $k \geq 1$.

- With $k = 1$ the definition requires every entry of a TU matrix to be 0, +1 or -1.

Theorem 4.14: the incidence matrix is totally unimodular

The node-arc incidence matrix $\mathbf{B}$ of every directed graph $G = (N, A)$ is totally unimodular.

Proof. Let $\mathbf{M}$ be a $k \times k$ submatrix of $\mathbf{B}$; we prove $\det(\mathbf{M}) \in \{0, +1, -1\}$ by induction on k . For $k = 1$ the entry is 0, +1 or -1. Let $k \geq 2$. Every column of $\mathbf{B}$ has exactly one +1 and one -1, so every column of $\mathbf{M}$ has at most two nonzero entries; there are three cases.

1. *A column of $\mathbf{M}$ is zero.* Then $\det(\mathbf{M}) = 0$.
2. *A column of $\mathbf{M}$ has exactly one nonzero entry, ± 1 .* The Laplace expansion along this column gives $\det(\mathbf{M}) = \pm \det(\mathbf{M}')$, where $\mathbf{M}'$ is a $(k-1) \times (k-1)$ submatrix of $\mathbf{B}$: by induction $\det(\mathbf{M}') \in \{0, +1, -1\}$.
3. *Every column of $\mathbf{M}$ has two nonzero entries, +1 and -1.* Then the rows of $\mathbf{M}$ add up to the zero vector, they are linearly dependent, and $\det(\mathbf{M}) = 0$. $\square$

Example 4.6: submatrices of the reference incidence matrix

We take rows 2, 3, 4 of the incidence matrix $\mathbf{B}$ of the reference graph and two choices of three columns.

	a_1	a_3	a_5		a_3	a_4	a_6
2	-1	1	0	2	1	-1	0
3	0	0	1	3	0	1	-1
4	0	-1	0	4	-1	0	1

- The column of $a_1 = (1, 2)$ in the first matrix has one nonzero entry (case 2): expanding along it, and then along the column of a_5 , the determinant is -1.
- The arcs $a_3 = (2, 4)$, $a_4 = (3, 2)$, $a_6 = (4, 3)$ of the second matrix form the cycle 2, 4, 3, 2, and all their endpoints are among the chosen rows: every column has +1 and -1 (case 3), and the determinant is 0.

- The matrix B has 5004 square submatrices; computing all their determinants confirms that they take only the values -1 , 0 and $+1$.

4.6.1 Integer vertices

- A point $\bar{x}$ of a polyhedron P is a **vertex** if it is not the midpoint of two distinct points of P , as in [Chapter 15 «The geometry of Linear Programming»](#) of the module Linear Programming.
- For a set of arcs F , we denote by B_F the submatrix of the columns of B , and by x_F the corresponding entries of x .

Theorem 4.15: integer vertices

Let $G = (N, A)$ be a network with integer balances b , integer lower bounds l and capacities u that are integer or $+\infty$. Then every vertex of

$$P = \{ x \in \mathbb{R}^m : Bx = b, l \leq x \leq u \}$$

is an integer vector.

Proof. Let $\bar{x}$ be a vertex of P , and let $F = \{ a \in A : l_a < \bar{x}_a < u_a \}$ be the arcs strictly between their bounds. If $F = \emptyset$, every entry of $\bar{x}$ is a bound, hence an integer.

1. *The columns of B_F are linearly independent.* Otherwise $B_F d_F = 0$ for some $d \neq 0$ with $d_a = 0$ for $a \notin F$. For a small $\varepsilon > 0$ both $\bar{x} + \varepsilon d$ and $\bar{x} - \varepsilon d$ satisfy the strict bounds on F and $Bx = b$, so they lie in P , and $\bar{x}$ is their midpoint: a contradiction.
2. *The system for $\bar{x}_F$ has integer data.* Every $a \notin F$ has $\bar{x}_a \in \{l_a, u_a\}$, an integer, so $B_F \bar{x}_F = b - \sum_{a \notin F} B_a \bar{x}_a =: \hat{b}$ is integer.
3. *Cramer's rule.* B_F has rank $|F|$, so some $|F|$ of its rows form a nonsingular square matrix M , and $M \bar{x}_F = \hat{b}_R$ on these rows R . By Cramer's rule each entry of $\bar{x}_F$ is $\det(M_a) / \det(M)$, with M_a an integer matrix.
4. By [Theorem 4.14](#), $\det(M) = \pm 1$, so every entry of $\bar{x}_F$ is an integer. $\square$

Corollary 4.16: integrality of the minimum cost flow model

Let the balances, the lower bounds and the capacities be integer, with every capacity finite. If the MCF model has a feasible solution, then it has an integer optimal solution, for any costs c .

Proof. The polyhedron P is nonempty and bounded, since $l \leq x \leq u$ with finite bounds. A linear function attains its minimum over a nonempty bounded polyhedron at a vertex ([Chapter 15 «The geometry of Linear Programming»](#)), and every vertex of P is integer by [Theorem 4.15](#). $\square$

- The simplex algorithm ends at a vertex, so with integer data it returns an integer optimal flow: the integer requirement $x_{ij} \in \mathbb{Z}_{\geq 0}$ comes for free.
- Total unimodularity is a property of the matrix: for a matrix that is not TU, an integer right-hand side can give fractional vertices.

Example 4.7: a matrix that is not totally unimodular

The undirected triangle with vertices 1, 2, 3 and edges $e_1 = \{1, 2\}$, $e_2 = \{1, 3\}$, $e_3 = \{2, 3\}$ has the vertex–edge incidence matrix

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \quad \det(\mathbf{M}) = -2,$$

so $\mathbf{M}$ is not TU. The polyhedron $\{\mathbf{y} \in \mathbb{R}^3 : \mathbf{M}\mathbf{y} = \mathbf{1}, \mathbf{y} \geq \mathbf{0}\}$ contains the single point $\mathbf{y} = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, since $\mathbf{M}$ is nonsingular: this vertex is fractional, although the data are integer.

The incidence matrix of a directed graph is totally unimodular. Hence, with integer balances and bounds, every vertex of the feasible region is integer, and the MCF model has an integer optimal solution.

4.7 The dual model and the reduced costs

- By [Proposition 4.9](#) the lower bounds can be removed, so in this section $\mathbf{l} = \mathbf{0}$ and every capacity is finite. The constraint $x_{ij} \leq u_{ij}$ is written as $-x_{ij} \geq -u_{ij}$.
- Following the rules of [Chapter 9 «Duality theory»](#) of the module Linear Programming, we use a free dual variable π_i , the **potential** of node i , for its conservation equation, and a non-negative dual variable α_{ij} for the capacity of (i, j) .

Using free dual variables π_i for the constraints (4.1b) and non-negative dual variables α_{ij} for the capacities, the dual model of the MCF model with $\mathbf{l} = \mathbf{0}$ is:

$$\max \sum_{i \in N} b_i \pi_i - \sum_{(i,j) \in A} u_{ij} \alpha_{ij} \quad (4.2a)$$

$$\text{subject to } \pi_i - \pi_j - \alpha_{ij} \leq c_{ij}, \quad \forall (i, j) \in A, \quad (4.2b)$$

$$\pi_i \in \mathbb{R}, \quad \forall i \in N, \quad (4.2c)$$

$$\alpha_{ij} \geq 0, \quad \forall (i, j) \in A. \quad (4.2d)$$

This LP model is referred to as the $D(\text{MCF})$ model.

- Description of the dual model:
 - the column of x_{ij} has +1 in the equation of i , -1 in the equation of j and -1 in its capacity constraint, which gives the dual constraint (4.2b), one for every arc;

- the primal variables are non-negative, so the dual constraints are of type $\leq$.
- Adding the same constant to all the potentials changes neither (4.2b) nor the objective, since $\sum_{i \in N} b_i = 0$: the potentials can be normalized, for example with $\pi_n = 0$.

Definition 4.17: reduced cost

Given potentials $\pi \in \mathbb{R}^n$, the **reduced cost** of the arc $(i, j) \in A$ is

$$\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j.$$

- The dual constraint (4.2b) reads $\bar{c}_{ij} + \alpha_{ij} \geq 0$. For fixed potentials, the best choice of $\alpha_{ij} \geq 0$ is the smallest one, $\alpha_{ij} = \max\{0, -\bar{c}_{ij}\}$.

Lemma 4.18: cost and reduced cost

For every x with $Bx = b$ and every $\pi \in \mathbb{R}^n$,

$$\sum_{(i,j) \in A} c_{ij} x_{ij} = \sum_{(i,j) \in A} \bar{c}_{ij} x_{ij} + \sum_{i \in N} b_i \pi_i.$$

Proof. Since $c_{ij} = \bar{c}_{ij} + \pi_i - \pi_j$, it suffices to show $\sum_{(i,j) \in A} (\pi_i - \pi_j) x_{ij} = \sum_{i \in N} b_i \pi_i$. The term $\pi_i x_{ij}$ goes to the tail and $-\pi_j x_{ij}$ to the head, so

$$\sum_{(i,j) \in A} (\pi_i - \pi_j) x_{ij} = \sum_{i \in N} \pi_i \left(\sum_{(i,j) \in \delta^+(i)} x_{ij} - \sum_{(j,i) \in \delta^-(i)} x_{ji} \right) = \sum_{i \in N} \pi_i b_i. \quad \square$$

Theorem 4.19: duality gap

Let x be a feasible flow ($l = 0$) and (π, α) a feasible solution of the $D(\text{MCF})$ model. Then

$$\sum_{(i,j) \in A} c_{ij} x_{ij} - \left(\sum_{i \in N} b_i \pi_i - \sum_{(i,j) \in A} u_{ij} \alpha_{ij} \right) = \sum_{(i,j) \in A} (\bar{c}_{ij} + \alpha_{ij}) x_{ij} + \sum_{(i,j) \in A} \alpha_{ij} (u_{ij} - x_{ij}) \geq 0,$$

and both solutions are optimal if and only if, for every arc, $(\bar{c}_{ij} + \alpha_{ij}) x_{ij} = 0$ and $\alpha_{ij} (u_{ij} - x_{ij}) = 0$.

Proof. By Lemma 4.18, the left-hand side is $\sum \bar{c}_{ij} x_{ij} + \sum u_{ij} \alpha_{ij}$, and $u_{ij} \alpha_{ij} = \alpha_{ij} x_{ij} + \alpha_{ij} (u_{ij} - x_{ij})$ gives the identity.

- Every term is non-negative: $\bar{c}_{ij} + \alpha_{ij} \geq 0$ by (4.2b), $\alpha_{ij} \geq 0$, and $0 \leq x_{ij} \leq u_{ij}$.
- If the gap is zero, the primal cost equals a dual value, which is a lower bound on every primal cost by the inequality just proved: x is optimal, and in the same way (π, α) is optimal.
- If both are optimal, the gap is zero by the strong duality theorem of Chapter 9 «Duality theory». A sum of non-negative terms is zero if and only if every term is zero. $\square$

Corollary 4.20: reduced cost certificate

Let $\mathbf{x}$ be a feasible flow ($\mathbf{l} = \mathbf{0}$) and $\boldsymbol{\pi} \in \mathbb{R}^n$ potentials such that, for every arc,

$$\bar{c}_{ij} > 0 \Rightarrow x_{ij} = 0, \quad \bar{c}_{ij} < 0 \Rightarrow x_{ij} = u_{ij}.$$

Then $\mathbf{x}$ is an optimal flow, and $\boldsymbol{\pi}$ with $\alpha_{ij} = \max\{0, -\bar{c}_{ij}\}$ is an optimal solution of the $D(\text{MCF})$ model.

Proof. The pair $(\boldsymbol{\pi}, \boldsymbol{\alpha})$ is dual feasible, since $\alpha_{ij} \geq 0$ and $\bar{c}_{ij} + \alpha_{ij} \geq 0$. We check the two conditions of [Theorem 4.19](#) on every arc.

- If $\bar{c}_{ij} \geq 0$, then $\alpha_{ij} = 0$, and $\bar{c}_{ij}x_{ij} = 0$ because $\bar{c}_{ij} > 0$ forces $x_{ij} = 0$.
- If $\bar{c}_{ij} < 0$, then $\bar{c}_{ij} + \alpha_{ij} = 0$, and $\alpha_{ij}(u_{ij} - x_{ij}) = 0$ because $x_{ij} = u_{ij}$.

The gap is zero, so both solutions are optimal. □

- An arc with positive reduced cost is empty, an arc with negative reduced cost is saturated, and an arc strictly between its bounds has reduced cost zero.

Example 4.8: the dual of the reference network

For the network of [Example 4.1](#), the $D(\text{MCF})$ model is

$$\max \quad 5\pi_1 + 2\pi_2 - 3\pi_4 - 4\pi_6 - (4\alpha_{12} + 4\alpha_{13} + 4\alpha_{24} + 2\alpha_{32} + 4\alpha_{35} + 3\alpha_{43} + 5\alpha_{46} + 3\alpha_{54} + 2\alpha_{56})$$

with $\boldsymbol{\alpha} \geq \mathbf{0}$, free potentials, and one constraint per arc:

$$\begin{aligned} \pi_1 - \pi_2 - \alpha_{12} &\leq 2, & \pi_1 - \pi_3 - \alpha_{13} &\leq 2, & \pi_2 - \pi_4 - \alpha_{24} &\leq 5, \\ \pi_3 - \pi_2 - \alpha_{32} &\leq 2, & \pi_3 - \pi_5 - \alpha_{35} &\leq 3, & \pi_4 - \pi_3 - \alpha_{43} &\leq 3, \\ \pi_4 - \pi_6 - \alpha_{46} &\leq 5, & \pi_5 - \pi_4 - \alpha_{54} &\leq 5, & \pi_5 - \pi_6 - \alpha_{56} &\leq 3. \end{aligned}$$

The balances $b_3 = b_5 = 0$ make π_3 and π_5 disappear from the objective.

- The potentials and the reduced costs are the basis of the optimality tests of [Chapter 6](#) «Minimum cost flow».

The dual of the MCF model has a potential π_i per node; the reduced cost is $\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j$. A feasible flow whose positive-reduced-cost arcs are empty and negative-reduced-cost arcs are saturated is optimal.

4.8 Special cases

- Several classical problems are MCF models with special data; all the lower bounds are zero.

problem	the MCF model
Shortest path	$b_s = 1, b_t = -1, b_i = 0$ otherwise; $u_{ij} = +\infty, c_{ij}$ the length of the arc: the unit of flow travels from s to t .
Maximum flow	$b_i = 0$ and $c_{ij} = 0$ for every arc, plus the arc (t, s) with $c_{ts} = -1$ and $u_{ts} = +\infty$: the flow on (t, s) is the value sent from s to t .
Transportation	$N = P \cup D$ with arcs from the origins P to the destinations D ; b_i the supply of $i \in P, -b_j$ the demand of $j \in D, u_{ij} = +\infty$.
Assignment	a transportation model with $ P = D , b_i = 1$ for $i \in P$ and $b_j = -1$ for $j \in D$.

- In the four models the balances and the bounds are integer, so the integer vertices of [Theorem 4.15](#) are paths, integer flows and assignments.

Shortest path, maximum flow, transportation and assignment are minimum cost flow models with special balances, capacities and costs.

4.9 A transshipment instance

- We solve the MCF model of [Example 4.2](#): the reference network, with two supply nodes, two demand nodes and two transshipment nodes. The optimal solution is computed with Gurobi (gurobipy), and then certified by hand with [Corollary 4.20](#).

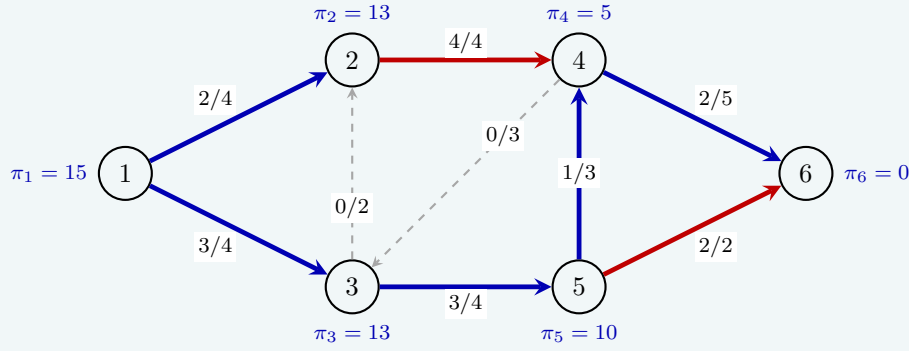
Example 4.9: the optimal flow

The optimal flow $\tilde{x}$ and its optimal value are

arc	(1, 2)	(1, 3)	(2, 4)	(3, 2)	(3, 5)	(4, 3)	(4, 6)	(5, 4)	(5, 6)
$\tilde{x}_{ij}$	2	3	4	0	3	0	2	1	2
u_{ij}	4	4	4	2	4	3	5	3	2

$z(\text{MCF}) = 60.$

In the figure each arc carries $\tilde{x}_{ij}/u_{ij}$: the arcs strictly between their bounds are blue, the arcs at capacity red, the empty arcs grey and dashed. Next to each node, its potential.



The flow is integer, as [Corollary 4.16](#) guarantees.

4.9.1 The certificate of optimality

1. The five blue arcs $(1,2)$, $(1,3)$, $(3,5)$, $(4,6)$, $(5,4)$ form a spanning tree. We ask for reduced cost zero on them, $\pi_i - \pi_j = c_{ij}$, and normalize with $\pi_6 = 0$.
2. Going backwards from node 6: $\pi_4 = \pi_6 + c_{46} = 5$, $\pi_5 = \pi_4 + c_{54} = 10$, $\pi_3 = \pi_5 + c_{35} = 13$, $\pi_1 = \pi_3 + c_{13} = 15$ and $\pi_2 = \pi_1 - c_{12} = 13$.
3. The potential π_i is the cost of the tree path from i to 6: for example, $\pi_1 = c_{13} + c_{35} + c_{54} + c_{46} = 2 + 3 + 5 + 5 = 15$.

Example 4.10: reduced costs and optimality

The reduced costs $\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j$ are:

arc	(1,2)	(1,3)	(2,4)	(3,2)	(3,5)	(4,3)	(4,6)	(5,4)	(5,6)
$\bar{c}_{ij}$	0	0	-3	2	0	11	0	0	-7
α_{ij}	0	0	3	0	0	0	0	0	7

- The arcs with $\bar{c}_{ij} > 0$, $(3,2)$ and $(4,3)$, are empty; the arcs with $\bar{c}_{ij} < 0$, $(2,4)$ and $(5,6)$, are at capacity. By [Corollary 4.20](#), $\tilde{\mathbf{x}}$ is optimal.
- The dual value is $\sum_{i \in N} b_i \pi_i - \sum_{(i,j) \in A} u_{ij} \alpha_{ij} = 86 - 26 = 60$, equal to the cost of $\tilde{\mathbf{x}}$. The potentials coincide with the dual values returned by Gurobi, shifted so that $\pi_6 = 0$.

- **Uniqueness.** Every optimal flow has gap zero with $(\boldsymbol{\pi}, \boldsymbol{\alpha})$, so by [Theorem 4.19](#) it has $x_{32} = x_{43} = 0$, $x_{24} = 4$ and $x_{56} = 2$. The flow on the five tree arcs is then fixed by conservation, since their columns are linearly independent: $\tilde{\mathbf{x}}$ is the unique optimal flow.
- **The flow $\tilde{\mathbf{x}}$.** The feasible flow $\tilde{\mathbf{x}}$ of [Example 4.1](#) costs 63, and its gap with the same dual solution is $63 - 60 = 3 = \alpha_{24}(u_{24} - \tilde{x}_{24}) = 3 \cdot (4 - 3)$: the arc $(2,4)$ is not used enough.

On the reference network the optimal flow costs 60. The potentials of its spanning tree give reduced costs that are positive only on empty arcs and negative only on saturated arcs: they certify the optimality without a solver.

4.10 Typical mistakes

Warning

- **Swapping outflow and inflow.** The conservation constraint is outflow minus inflow equal to b_i , with $b_i > 0$ at a supply node: the signs follow the incidence matrix, +1 in the tail.
- **Taking $\sum_i b_i = 0$ as sufficient.** A balanced network can have no feasible flow: a cut may be too small to carry the supply inside it.
- **Getting the substitution signs wrong.** With $\mathbf{x} = \mathbf{l} + \mathbf{x}'$ the tail of (i, j) loses l_{ij} units of balance and the head gains them; the constant $\mathbf{c}'\mathbf{l}$ must be added back to the value.
- **Forgetting the dummy node.** An instance with more supply than demand has no feasible flow with equalities; the dummy arcs must cost zero, or the unused supply is charged.
- **Claiming integrality for any matrix.** Integer vertices need a TU matrix and integer $\mathbf{b}, \mathbf{l}, \mathbf{u}$; the costs may be fractional.
- **Writing the reduced cost with the wrong signs.** In this module $\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j$, with $-\pi$ of the tail and $+\pi$ of the head.
- **Expecting unique potentials.** The potentials are defined up to an additive constant; they are fixed by a normalization such as $\pi_n = 0$.

4.11 In summary

Summary

- A network has balances b_i , lower bounds l_{ij} , capacities u_{ij} and costs c_{ij} ; the minimum cost flow problem is the LP $\min\{\mathbf{c}'\mathbf{x} : \mathbf{B}\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$.
- Across every cut the net outflow equals the balance inside; a feasible network is balanced and satisfies the cut condition.
- Lower bounds are removed with $\mathbf{x} = \mathbf{l} + \mathbf{x}'$; an excess of supply goes to a dummy node through zero-cost arcs.
- The incidence matrix is totally unimodular, so with integer $\mathbf{b}, \mathbf{l}, \mathbf{u}$ every vertex is integer and the MCF model has an integer optimal solution.
- The dual has potentials π_i ; with $\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j$, a flow with empty positive-cost

arcs and saturated negative-cost arcs is optimal.

- Shortest path, maximum flow, transportation and assignment are MCF models.

The reference network at a glance.

balances	$\mathbf{b} = (5, 2, 0, -3, 0, -4)$
optimal flow	$\tilde{\mathbf{x}} = (2, 3, 4, 0, 3, 0, 2, 1, 2), z(\text{MCF}) = 60$
potentials ($\pi_6 = 0$)	$\boldsymbol{\pi} = (15, 13, 13, 5, 10, 0)$
empty / saturated arcs	$(3, 2), (4, 3) / (2, 4), (5, 6)$
with $l_{32} = 1$	$\mathbf{b}' = (5, 3, -1, -3, 0, -4)$, optimal value 62
with $b_2 = 4$ (dummy node 7)	$x_{17} = 2$, optimal value 56

Maximum flow

Prerequisites

- From this module, [Chapter 1 «Graphs»](#): directed paths, the breadth-first search and the depth-first search.
- From this module, [Chapter 4 «Network flow models»](#): flow conservation, the flow across a cut, the minimum cost flow model and its integer vertices.
- From the module Linear Programming, [Chapter 15 «The geometry of Linear Programming»](#): a linear function attains its maximum over a nonempty bounded polyhedron.

Objectives

At the end of this chapter you will be able to:

- state the maximum flow problem and write its LP model;
- compute the capacity of an s - t cut, and use cuts as upper bounds on the value of every flow;
- build the residual network of a flow, find an augmenting path and its bottleneck, and augment the flow;
- prove the max-flow min-cut theorem and the integrality theorem;
- run the Ford–Fulkerson and the Edmonds–Karp algorithms, read a minimum cut at the stop, and give their running times;
- compute a maximum matching in a bipartite graph and a maximum set of arc-disjoint paths as maximum flows.

5.1 The maximum flow problem

Definition 5.1: maximum flow problem

Let $G = (N, A)$ be a directed graph with $n = |N|$ nodes and $m = |A|$ arcs, two distinct nodes s , the **source**, and t , the **sink**, and a **capacity** $u_{ij} \geq 0$ for every arc $(i, j) \in A$.

- A **feasible flow** is a vector $\mathbf{x} \in \mathbb{R}^m$ with $0 \leq x_{ij} \leq u_{ij}$ for every arc $(i, j) \in A$ and flow conservation at every node other than s and t :

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} - \sum_{(j,i) \in \delta^-(i)} x_{ji} = 0, \quad \forall i \in N \setminus \{s, t\}.$$

- The **value** of the flow is the net outflow of the source:

$$v = \sum_{(s,j) \in \delta^+(s)} x_{sj} - \sum_{(j,s) \in \delta^-(s)} x_{js}.$$

- The **maximum flow problem** asks for a feasible flow of maximum value, a **maximum flow**. Its value is denoted by $\tilde{v}$.

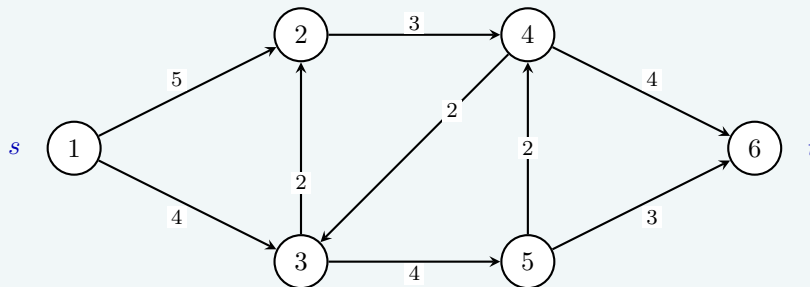
- The capacities are finite, and the flow $\mathbf{x} = \mathbf{0}$, of value 0, is always feasible.
- The nodes other than s and t pass on everything they receive: the flow leaves s and travels through the network towards t .
- As in Chapter 4 «Network flow models», $\delta^+(i)$ and $\delta^-(i)$ are the arcs leaving and entering node i , and the arcs are numbered $a_1, a_2, \dots, a_m$ in lexicographic order.

Example 5.1: the reference network

We give capacities to the reference graph of Chapter 1 «Graphs», with source $s = 1$ and sink $t = 6$:

arc	(1, 2)	(1, 3)	(2, 4)	(3, 2)	(3, 5)	(4, 3)	(4, 6)	(5, 4)	(5, 6)
u_{ij}	5	4	3	2	4	2	4	2	3
$\bar{x}_{ij}$	3	0	3	0	2	2	1	0	2

In the figure each arc carries its capacity u_{ij} .

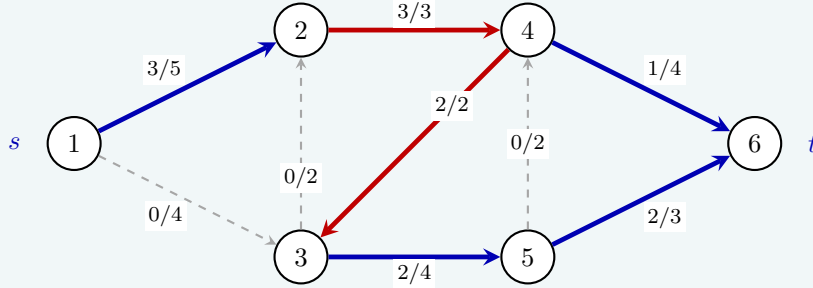


The flow $\bar{\mathbf{x}}$ of the table is feasible. For example, at node 4 the inflow is $\bar{x}_{24} + \bar{x}_{54} = 3 + 0 = 3$

and the outflow is $\bar{x}_{43} + \bar{x}_{46} = 2 + 1 = 3$. Its value is

$$v = \bar{x}_{12} + \bar{x}_{13} = 3 + 0 = 3,$$

and node 6 receives $\bar{x}_{46} + \bar{x}_{56} = 1 + 2 = 3$ units. In the figure each arc carries $\bar{x}_{ij}/u_{ij}$: the arcs with $0 < \bar{x}_{ij} < u_{ij}$ are blue, the saturated arcs ($\bar{x}_{ij} = u_{ij}$) red, and the empty arcs grey and dashed.



5.1.1 The LP model

- We use one continuous variable x_{ij} for every arc $(i, j) \in A$, the flow on the arc, and one variable v , the value of the flow.

Using these variables, an LP model for the maximum flow problem reads as follows:

$$\max v \quad (5.1a)$$

$$\text{subject to } \sum_{(s,j) \in \delta^+(s)} x_{sj} - \sum_{(j,s) \in \delta^-(s)} x_{js} = v, \quad (5.1b)$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} - \sum_{(j,i) \in \delta^-(i)} x_{ji} = 0, \quad \forall i \in N \setminus \{s, t\}, \quad (5.1c)$$

$$\sum_{(t,j) \in \delta^+(t)} x_{tj} - \sum_{(j,t) \in \delta^-(t)} x_{jt} = -v, \quad (5.1d)$$

$$x_{ij} \leq u_{ij}, \quad \forall (i, j) \in A, \quad (5.1e)$$

$$x_{ij} \geq 0, \quad \forall (i, j) \in A, \quad (5.1f)$$

$$v \in \mathbb{R}. \quad (5.1g)$$

This LP model is referred to as the MF model.

- Description of the objective function and constraints of the MF model:
 - the objective function (5.1a) maximizes the value of the flow;
 - constraint (5.1b) states that the net outflow of s is v , and constraint (5.1d) that the net inflow of t is v ;
 - constraints (5.1c) are the flow conservation constraints of the other nodes;

- constraints (5.1e) impose the capacities, and constraints (5.1f) and (5.1g) define the variables of the LP model.
- Every arc contributes $+x_{ij}$ to the row of its tail and $-x_{ij}$ to the row of its head, so the left-hand sides of (5.1b)–(5.1d) add up to zero: constraint (5.1d) follows from the others.
- In Chapter 4, Section «Special cases» the maximum flow problem is a minimum cost flow model: zero balances and zero costs, plus the arc (t, s) with capacity $+\infty$ and cost -1 . Calling v the flow x_{ts} on this arc, (5.1b)–(5.1d) are its conservation constraints, and $\min(-x_{ts})$ is $\max v$.

A feasible flow respects the capacities and is conserved at every node other than s and t ; its value v is the net outflow of s . The maximum flow problem is the LP MF: maximize v under conservation and capacities.

5.2 Cuts

Definition 5.2: s – t cut

Let $S \subseteq N$ with $s \in S$ and $t \notin S$. The s – t **cut** $(S, N \setminus S)$ is the partition of the nodes into S and $N \setminus S$.

- Its **forward arcs** are the arcs of $\delta^+(S)$, from S to $N \setminus S$, and its **backward arcs** the arcs of $\delta^-(S)$, from $N \setminus S$ to S .
- Its **capacity** is the total capacity of the forward arcs:

$$u(S, N \setminus S) = \sum_{(i,j) \in \delta^+(S)} u_{ij}.$$

- A **minimum cut** is an s – t cut of minimum capacity.

- The capacities of the backward arcs do not enter $u(S, N \setminus S)$.
- Each node other than s and t can be on either side, so a network has 2^{n-2} s – t cuts.
- As in Chapter 4, $x(F) = \sum_{(i,j) \in F} x_{ij}$ for a set of arcs F .

Lemma 5.3: flow across an s – t cut

Let $\mathbf{x}$ be a feasible flow of value v . Then, for every s – t cut $(S, N \setminus S)$,

$$v = x(\delta^+(S)) - x(\delta^-(S)).$$

In particular, with $S = N \setminus \{t\}$, the net inflow of t is v .

Proof. Let b_i be the net outflow of node i under $\mathbf{x}$: $b_s = v$ and $b_i = 0$ for $i \notin \{s, t\}$. Then $\mathbf{x}$ satisfies flow conservation with these balances, and Chapter 4, Lemma «flow across a cut» gives $x(\delta^+(S)) - x(\delta^-(S)) = \sum_{i \in S} b_i$ for every $S \subseteq N$.

1. With $S = N$ no arc leaves or enters S , so $\sum_{i \in N} b_i = 0$ and $b_t = -v$.
2. For an s - t cut, $s \in S$ and $t \notin S$, so $\sum_{i \in S} b_i = b_s = v$. □

Corollary 5.4: weak duality

For every feasible flow $\mathbf{x}$ of value v and every s - t cut $(S, N \setminus S)$,

$$v \leq u(S, N \setminus S),$$

with equality if and only if $x_{ij} = u_{ij}$ for every $(i, j) \in \delta^+(S)$ and $x_{ij} = 0$ for every $(i, j) \in \delta^-(S)$. In case of equality, $\mathbf{x}$ is a maximum flow and $(S, N \setminus S)$ is a minimum cut.

Proof. By Lemma 5.3 and the bounds $x_{ij} \leq u_{ij}$ on $\delta^+(S)$ and $x_{ij} \geq 0$ on $\delta^-(S)$,

$$v = x(\delta^+(S)) - x(\delta^-(S)) \leq u(S, N \setminus S) - 0.$$

- Equality holds if and only if every bound used is tight, arc by arc.
- In case of equality, every feasible flow $\mathbf{x}'$ has value $v' \leq u(S, N \setminus S) = v$, and every s - t cut $(S', N \setminus S')$ has $u(S', N \setminus S') \geq v = u(S, N \setminus S)$. □

Example 5.2: the cuts of the reference network

The reference network of Example 5.1 has $2^4 = 16$ s - t cuts. Next to each set S , the capacity $u(S, N \setminus S)$:

S	u	S	u	S	u	S	u
$\{1\}$	9	$\{1, 2\}$	7	$\{1, 3\}$	11	$\{1, 4\}$	15
$\{1, 5\}$	14	$\{1, 2, 3\}$	7	$\{1, 2, 4\}$	10	$\{1, 2, 5\}$	12
$\{1, 3, 4\}$	15	$\{1, 3, 5\}$	12	$\{1, 4, 5\}$	18	$\{1, 2, 3, 4\}$	8
$\{1, 2, 3, 5\}$	8	$\{1, 2, 4, 5\}$	13	$\{1, 3, 4, 5\}$	14	$\{1, 2, 3, 4, 5\}$	7

- The minimum capacity is 7, attained by three cuts: $S = \{1, 2\}$, $S = \{1, 2, 3\}$ and $S = \{1, 2, 3, 4, 5\}$. By Corollary 5.4, no feasible flow has value larger than 7.
- For $S = \{1, 2, 3\}$ the forward arcs are $(2, 4)$, $(3, 5)$ and the backward arc is $(4, 3)$. With the flow $\bar{\mathbf{x}}$ of Example 5.1,

$$\bar{x}_{24} + \bar{x}_{35} - \bar{x}_{43} = 3 + 2 - 2 = 3 = v, \quad u(S, N \setminus S) = u_{24} + u_{35} = 3 + 4 = 7.$$

The capacity of a cut

The capacity $u(S, N \setminus S)$ adds up the capacities of the forward arcs only. For $S = \{1, 2, 3\}$ it is $u_{24} + u_{35} = 7$; adding the capacity of the backward arc $(4, 3)$ would give 9, which bounds nothing. A flow on a backward arc brings units back into S and lowers the net flow

across the cut.

The net flow across every s - t cut is the value v , so $v \leq u(S, N \setminus S)$ for every flow and every cut, with equality when the forward arcs are saturated and the backward arcs empty. On the reference network no flow has value larger than 7.

5.3 The residual network

- In the rest of the chapter G has no pair of opposite arcs (i, j) and (j, i) , as the reference graph. An arc (j, i) opposite to (i, j) can be replaced by the arcs (j, k) and (k, i) through a new node k , both with capacity u_{ji} : conservation at k gives $x_{jk} = x_{ki}$, and the flows of the two networks correspond, with the same value.

Definition 5.5: residual network

Let $\mathbf{x}$ be a feasible flow. The **residual network** $G(\mathbf{x}) = (N, A(\mathbf{x}))$ has, for every arc $(i, j) \in A$:

- the **forward arc** (i, j) , with **residual capacity** $r_{ij} = u_{ij} - x_{ij}$, if $x_{ij} < u_{ij}$;
- the **backward arc** (j, i) , with residual capacity $r_{ji} = x_{ij}$, if $x_{ij} > 0$.

- The residual capacity r_{ij} is how much flow can still be sent from i to j : by raising x_{ij} on a forward arc, or by lowering x_{ji} on a backward arc.
- An empty arc gives only its forward arc, a saturated arc only its backward arc, and an arc with $0 < x_{ij} < u_{ij}$ both. Hence $G(\mathbf{x})$ has at most $2m$ arcs, all with positive residual capacity.
- Without opposite arcs in G , every ordered pair of nodes carries at most one residual arc.

Definition 5.6: augmenting path

Let $\mathbf{x}$ be a feasible flow. An **augmenting path** is a directed path P from s to t in $G(\mathbf{x})$. Its **bottleneck** is

$$\delta = \min_{(i,j) \in P} r_{ij} > 0.$$

Augmenting $\mathbf{x}$ along P means sending δ units along P : for every arc of P ,

- if it is a forward arc (i, j) , then $x_{ij} := x_{ij} + \delta$;
- if it is a backward arc (j, i) , with $(i, j) \in A$, then $x_{ij} := x_{ij} - \delta$.

Lemma 5.7: augmentation

Augmenting a feasible flow $\mathbf{x}$ of value v along an augmenting path P with bottleneck δ gives a feasible flow $\mathbf{x}'$ of value $v + \delta$.

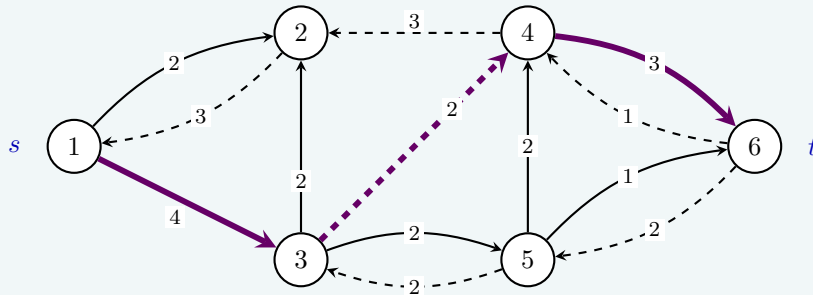
Proof. The nodes of P are distinct and G has no opposite arcs, so every arc of A changes at most once.

1. *Bounds.* On a forward arc, $x_{ij} + \delta \leq x_{ij} + r_{ij} = u_{ij}$; on a backward arc, $x_{ij} - \delta \geq x_{ij} - r_{ji} = 0$.
2. *Conservation.* Let $k \notin \{s, t\}$ be a node of P . The arc of P entering k raises the net inflow of k by δ : a forward arc (h, k) raises x_{hk} , a backward arc (h, k) lowers x_{kh} . In the same way, the arc of P leaving k raises the net outflow of k by δ . The two changes cancel, and the nodes outside P are not touched.
3. *Value.* The path P leaves s once and never enters it again, so the net outflow of s grows by δ . $\square$

Example 5.3: the residual network of $\bar{x}$

For the flow $\bar{x}$ of Example 5.1, of value 3, the residual network $G(\bar{x})$ has 13 arcs. The six backward arcs, dashed in the figure, are $(2, 1)$, $(3, 4)$, $(4, 2)$, $(5, 3)$, $(6, 4)$, $(6, 5)$.

- The arc $(2, 4)$ is saturated, so it gives only the backward arc $(4, 2)$, with $r_{42} = \bar{x}_{24} = 3$.
- The arc $(1, 2)$ has $0 < \bar{x}_{12} = 3 < 5$, so it gives the forward arc $(1, 2)$, with $r_{12} = 2$, and the backward arc $(2, 1)$, with $r_{21} = 3$.



The depth-first search of Chapter 1 from node 1, with the residual arcs of each node by increasing index of the head, labels the nodes 1, 2, 3, 4, 6: from 2 the only residual arc goes back to 1, from 3 the search takes the backward arc $(3, 4)$ (2 is labelled), and from 4 the arc $(4, 6)$. The augmenting path is $P = 1, 3, 4, 6$ (violet), with bottleneck

$$\delta = \min\{r_{13}, r_{34}, r_{46}\} = \min\{4, 2, 3\} = 2.$$

Augmenting gives $x_{13} = 0 + 2 = 2$, $x_{43} = 2 - 2 = 0$ (backward arc) and $x_{46} = 1 + 2 = 3$, with value $3 + 2 = 5$. The two units that went from 4 to 3 now go from 4 to 6, and node 3 receives two new units from 1, which continue along 3, 5, 6.

Example 5.4: why the backward arcs are needed

Suppose that from $\bar{x}$ we use only the forward arcs of the residual network, so that flows can only grow.

- The depth-first search on the forward arcs finds the path 1, 3, 5, 4, 6, with $\delta = \min\{4, 2, 2, 3\} = 2$, and the value becomes 5 with the flow $x = (3, 2, 3, 0, 4, 2, 3, 2, 2)$.
- Now the forward arcs (2, 4) and (3, 5) leaving $\{1, 2, 3\}$ are saturated: the search labels only 1, 2, 3, and the method stops with value 5.
- The value is not maximum: for $S = \{1, 2, 3\}$, Lemma 5.3 gives $v = x(\delta^+(S)) - x(\delta^-(S)) = 7 - 2 = 5$. The two units on the backward arc (4, 3) of the cut, sent there by the first choices, come back into S .

Only the backward arc (3, 4) of the residual network can cancel them: from this flow, the augmenting paths 1, 3, 4, 5, 6 (with the backward arcs (3, 4) and (4, 5)) and 1, 3, 4, 6, with $\delta = 1$ each, reach the value 7.

The residual network has a forward arc where the flow can grow and a backward arc where it can shrink. Sending the bottleneck δ along a path from s to t of the residual network keeps the flow feasible and raises its value by δ ; the backward arcs undo earlier choices.

5.4 The max-flow min-cut theorem

Theorem 5.8: max-flow min-cut

Let x be a feasible flow of value v . The following statements are equivalent:

- (a) x is a maximum flow;
- (b) the residual network $G(x)$ has no augmenting path;
- (c) $v = u(S, N \setminus S)$ for some s - t cut $(S, N \setminus S)$.

Moreover, a maximum flow exists, and the maximum value of a flow equals the minimum capacity of an s - t cut:

$$\tilde{v} = \min \{ u(S, N \setminus S) : S \subseteq N, s \in S, t \notin S \}.$$

Proof. (a) implies (b). An augmenting path would give, by Lemma 5.7, a feasible flow of value $v + \delta > v$.

(b) implies (c). Let S be the set of the nodes reachable from s in $G(x)$; then $s \in S$ and, by (b), $t \notin S$.

1. If an arc $(i, j) \in \delta^+(S)$ had $x_{ij} < u_{ij}$, the forward arc (i, j) would be in $G(x)$, and j would be reachable: hence $x_{ij} = u_{ij}$.
2. If an arc $(i, j) \in \delta^-(S)$ had $x_{ij} > 0$, the backward arc (j, i) , with $j \in S$, would

be in $G(\mathbf{x})$, and i would be reachable: hence $x_{ij} = 0$.

3. By [Corollary 5.4](#), $v = u(S, N \setminus S)$.

(c) implies (a). By [Corollary 5.4](#).

Existence and equality. The feasible region of the MF model is nonempty ($\mathbf{x} = \mathbf{0}$, $v = 0$) and bounded ($0 \leq x_{ij} \leq u_{ij}$ and $|v| \leq \sum_{(i,j) \in A} u_{ij}$), so the model has an optimal solution ([Chapter 15 «The geometry of Linear Programming»](#)). For a maximum flow, (a) implies (c): some cut has capacity $\tilde{v}$, and by [Corollary 5.4](#) no cut has a smaller capacity. $\square$

- The proof of (b) implies (c) is constructive: at a maximum flow, the nodes reachable from s in the residual network form a minimum cut.
- A flow and a cut with $v = u(S, N \setminus S)$ certify each other: the cut proves that no flow is better, and the flow proves that no cut is smaller.

Example 5.5: the maximum flow of the reference network

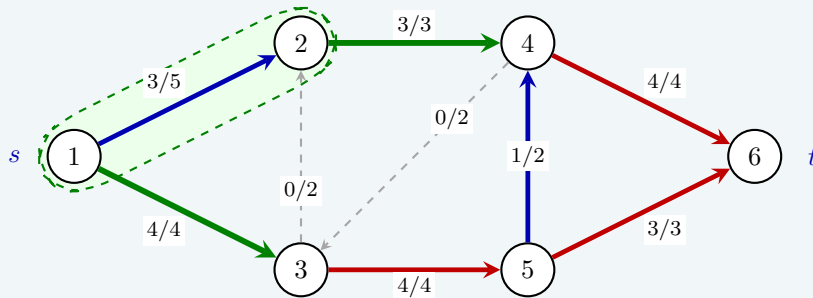
On the reference network of [Example 5.1](#) the flow

arc	(1, 2)	(1, 3)	(2, 4)	(3, 2)	(3, 5)	(4, 3)	(4, 6)	(5, 4)	(5, 6)
$\tilde{x}_{ij}$	3	4	3	0	4	0	4	1	3
u_{ij}	5	4	3	2	4	2	4	2	3

is feasible, with value $\tilde{x}_{12} + \tilde{x}_{13} = 7$. For $S = \{1, 2\}$ the forward arcs (1, 3), (2, 4) are saturated and the backward arc (3, 2) is empty, so

$$u(S, N \setminus S) = u_{13} + u_{24} = 4 + 3 = 7 = v.$$

By [Theorem 5.8](#), $\tilde{\mathbf{x}}$ is a maximum flow, $\tilde{v} = 7$, and $(\{1, 2\}, \{3, 4, 5, 6\})$ is a minimum cut. In $G(\tilde{\mathbf{x}})$ the nodes reachable from 1 are exactly 1 and 2. In the figure, S is shaded and the forward arcs of the cut are green.



- **Uniqueness.** By [Corollary 5.4](#), every maximum flow saturates the forward arcs and empties the backward arcs of each of the three minimum cuts of [Example 5.2](#):

$$S = \{1, 2\} : x_{13} = 4, x_{24} = 3, x_{32} = 0,$$

$$S = \{1, 2, 3\} : x_{35} = 4, x_{43} = 0,$$

$$S = \{1, 2, 3, 4, 5\} : x_{46} = 4, x_{56} = 3.$$

Conservation at nodes 1 and 4 then gives $x_{12} = 7 - x_{13} = 3$ and $x_{54} = x_{43} + x_{46} - x_{24} = 1$: $\tilde{\mathbf{x}}$ is the unique maximum flow, while the minimum cut is not unique.

Saturated arcs and cuts

The saturated arcs of a maximum flow do not form a cut in general. Under $\tilde{\mathbf{x}}$ the saturated arcs are $(1, 3)$, $(2, 4)$, $(3, 5)$, $(4, 6)$, $(5, 6)$, with total capacity 18. A minimum cut is read from the nodes reachable from s in the residual network.

A flow is maximum if and only if its residual network has no augmenting path, if and only if its value equals the capacity of a cut. The maximum value equals the minimum cut capacity, and the nodes reachable from s in the residual network of a maximum flow form a minimum cut.

5.5 The Ford–Fulkerson algorithm

- By [Theorem 5.8](#), a flow can be improved as long as its residual network has an augmenting path. The **Ford–Fulkerson algorithm** starts from the zero flow and augments along such paths until none is left.

Algorithm

Algorithm: Ford–Fulkerson algorithm

Input: a directed graph $G = (N, A)$ with integer capacities $u_{ij} \geq 0$, a source s and a sink t

Output: a maximum flow $\mathbf{x}$ and a minimum cut $(S, N \setminus S)$

- Step 0 (initialization).** Set $x_{ij} := 0$ for every $(i, j) \in A$, and $v := 0$.
- Step 1 (search).** Build the residual network $G(\mathbf{x})$ and search it from s .
- If t is not labelled, **STOP**: $\mathbf{x}$ is a maximum flow, and S is the set of the labelled nodes.
- Step 2 (augmentation).** Let P be the path of the predecessors from s to t , and $\delta := \min_{(i,j) \in P} r_{ij}$.
- Augment $\mathbf{x}$ along P by δ , set $v := v + \delta$, and go to Step 1.

- Step 1 can use any search of [Chapter 1, Section «Graph search»](#). In this module the Ford–Fulkerson algorithm uses the **depth-first search**: *LIST* is a stack, the residual arcs leaving each node are scanned by increasing index of the head, and the search stops as soon as t is labelled.
- When the search does not label t , it has labelled all the nodes reachable from s ([Chapter 1, Theorem «correctness of the search»](#)): they form the set S of Step 1.

Theorem 5.9: correctness of Ford–Fulkerson

Let the capacities be integer. Then every flow computed by the Ford–Fulkerson algorithm is integer, the algorithm stops after at most $\tilde{v}$ augmentations, and at the stop $\mathbf{x}$ is a maximum flow and $(S, N \setminus S)$ is a minimum cut.

Proof. We use [Lemma 5.7](#) and [Theorem 5.8](#).

1. *Integrality.* The zero flow is integer. If $\mathbf{x}$ is integer, all residual capacities are integer, so δ is a positive integer and the augmented flow is integer.
2. *Termination.* Every flow is feasible, and every augmentation raises the value by $\delta \geq 1$. No value exceeds $\tilde{v}$, so there are at most $\tilde{v}$ augmentations.
3. *Optimality.* At the stop t is not labelled, so $G(\mathbf{x})$ has no augmenting path and S is the set of the nodes reachable from s . The proof of (b) implies (c) gives $v = u(S, N \setminus S)$: the flow is maximum and the cut is minimum. $\square$

Corollary 5.10: integrality

If all the capacities are integer, there is a maximum flow with integer entries.

Proof. The flow returned by the Ford–Fulkerson algorithm, by [Theorem 5.9](#). $\square$

- The same conclusion follows from [Chapter 4, Theorem «integer vertices»](#): with the arc (t, s) , the MF model is a minimum cost flow model with integer data, and its optimum is attained at an integer vertex.
- Rational capacities become integer after multiplication by a common denominator, which multiplies every flow and every cut by the same factor: the algorithm stops also with rational capacities.

Theorem 5.11: running time of Ford–Fulkerson

With integer capacities, the Ford–Fulkerson algorithm runs in $O((n + m)\tilde{v})$ time, and $\tilde{v} \leq (n - 1)U$, where U is the largest capacity.

Proof. Every iteration builds $G(\mathbf{x})$ in $O(m)$ time, searches it in $O(n + m)$ time, since $G(\mathbf{x})$ has at most $2m$ arcs ([Chapter 1, Theorem «running time of the search»](#)), and changes at most $n - 1$ flows along P . There are at most $\tilde{v}$ iterations with an augmentation, plus the last one.

- By [Corollary 5.4](#) with $S = \{s\}$, $\tilde{v} \leq u(\{s\}, N \setminus \{s\})$, and s has at most $n - 1$ leaving arcs, each with capacity at most U . $\square$

- The bound grows with the value U of the capacities, and not only with the number of their digits, about $\log_2 U$: the running time is **pseudopolynomial**. [Example 5.7](#) shows that the number of augmentations can really grow with the capacities.

5.5.1 Ford–Fulkerson on the reference network

Example 5.6: Ford–Fulkerson on the reference network

We run the algorithm on the network of [Example 5.1](#). Each row is an iteration: the augmenting path, the residual capacities of its arcs, the backward arcs it uses, the bottleneck and the new value.

iteration	augmenting path	r_{ij} on P	backward arcs	δ	v
1	1, 2, 4, 3, 5, 6	5, 3, 2, 4, 3	–	2	2
2	1, 2, 4, 6	3, 1, 4	–	1	3
3	1, 3, 4, 6	4, 2, 3	(3, 4)	2	5
4	1, 3, 5, 4, 6	2, 2, 2, 1	–	1	6
5	1, 3, 5, 6	1, 1, 1	–	1	7

The flows after each iteration ($k = 0$ is the zero flow) are:

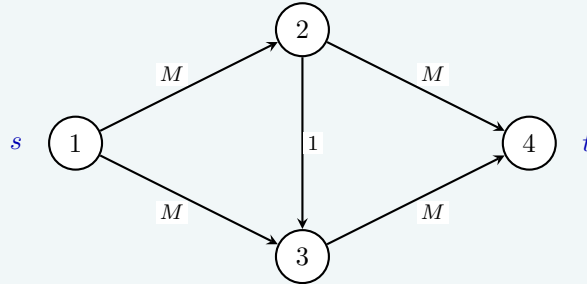
k	x_{12}	x_{13}	x_{24}	x_{32}	x_{35}	x_{43}	x_{46}	x_{54}	x_{56}
0	0	0	0	0	0	0	0	0	0
1	2	0	2	0	2	2	0	0	2
2	3	0	3	0	2	2	1	0	2
3	3	2	3	0	2	0	3	0	2
4	3	3	3	0	3	0	4	1	2
5	3	4	3	0	4	0	4	1	3

- Iteration 1 is the depth-first search of [Chapter 1, Example «DFS from node 1»](#): all the arcs are empty, $G(\mathbf{0}) = G$, and the path goes through the arc (4, 3).
- After iteration 2 the flow is $\bar{x}$ of [Example 5.1](#), and iteration 3 is the augmentation of [Example 5.3](#): the backward arc (3, 4) cancels the two units sent on (4, 3) in iteration 1.
- The sixth search labels only the nodes 1 and 2, and the algorithm stops: the flow is $\tilde{x}$ of [Example 5.5](#), and $S = \{1, 2\}$ gives the minimum cut of capacity $u_{13} + u_{24} = 7$.
- The algorithm makes 5 augmentations, within the bound $\tilde{v} = 7$ of [Theorem 5.9](#).

5.5.2 The number of augmentations

Example 5.7: many augmentations

In the network below, M is a positive integer. The paths 1, 2, 4 and 1, 3, 4 carry M units each, and the cut $S = \{1\}$ has capacity $2M$, so $\tilde{v} = 2M$.



- The Ford–Fulkerson algorithm allows any augmenting path. If the paths 1, 2, 3, 4 and 1, 3, 2, 4 alternate, the second one through the backward arc (3, 2), every bottleneck is $\delta = 1$, given by the arc (2, 3) or by its backward arc: $2M$ augmentations are needed, two million for $M = 10^6$.
- With the depth-first search of this module, for $M \geq 2$, the paths are 1, 2, 3, 4, then 1, 2, 4, 1, 3, 2, 4 and 1, 3, 4, with $\delta = 1$, $M - 1$, 1, $M - 1$: four augmentations.

The Ford–Fulkerson algorithm augments along paths of the residual network until none is left. With integer capacities it stops after at most $\tilde{v}$ augmentations, with an integer maximum flow and a minimum cut; its running time $O((n + m) \tilde{v})$ is pseudopolynomial.

5.6 The Edmonds–Karp algorithm

Definition 5.12: Edmonds–Karp algorithm

The **Edmonds–Karp algorithm** is the Ford–Fulkerson algorithm in which Step 1 uses the **breadth-first search**: *LIST* is a queue, the residual arcs leaving each node are scanned by increasing index of the head, and the search stops as soon as t is labelled. The capacities are any non-negative real numbers.

- By Chapter 1, Proposition «BFS finds shortest paths in number of arcs», the augmenting path has the minimum number of arcs among the paths from s to t in $G(\mathbf{x})$.
- For a feasible flow $\mathbf{x}$ and a node i , $d_{\mathbf{x}}(i)$ is the minimum number of arcs of a path from s to i in $G(\mathbf{x})$, and $d_{\mathbf{x}}(i) = +\infty$ if i is not reachable.
- An arc of the augmenting path is **critical** if its residual capacity equals δ : after the augmentation its residual capacity is zero, and it leaves the residual network.
- An augmentation along P lowers r_{ij} by δ and raises r_{ji} by δ on every arc (i, j) of P , and changes no other residual capacity. Hence the arcs that enter the residual network are reverses of arcs of P .

Lemma 5.13: the distances do not decrease

Let $\mathbf{x}'$ be obtained from $\mathbf{x}$ by an augmentation along a path P from s to t of $G(\mathbf{x})$ with the minimum number of arcs. Then $d_{\mathbf{x}'}(i) \geq d_{\mathbf{x}}(i)$ for every $i \in N$.

Proof. We show that every arc (a, b) of $G(\mathbf{x}')$ satisfies $d_{\mathbf{x}}(b) \leq d_{\mathbf{x}}(a) + 1$.

1. If (a, b) is an arc of $G(\mathbf{x})$, a shortest path from s to a followed by (a, b) reaches b with $d_{\mathbf{x}}(a) + 1$ arcs.
2. Otherwise (a, b) is the reverse of an arc (b, a) of P . The part of P from s to each of its nodes has the minimum number of arcs: a path to that node with fewer arcs, followed by the rest of P , would give a walk from s to t with fewer arcs than P , and a path inside it (Chapter 1, Lemma «a walk contains a path»). Hence $d_{\mathbf{x}}(a) = d_{\mathbf{x}}(b) + 1$, and $d_{\mathbf{x}}(b) \leq d_{\mathbf{x}}(a) + 1$.

Now let $s = i_0, i_1, \dots, i_q = i$ be a path of $G(\mathbf{x}')$ with $q = d_{\mathbf{x}'}(i)$ arcs. Along its arcs $d_{\mathbf{x}}$ grows by at most 1 at a time, so $d_{\mathbf{x}}(i) \leq d_{\mathbf{x}}(s) + q = d_{\mathbf{x}'}(i)$. If i is not reachable in $G(\mathbf{x}')$, then $d_{\mathbf{x}'}(i) = +\infty$. $\square$

Lemma 5.14: critical arcs

In the Edmonds–Karp algorithm, let the residual arc (i, j) be critical in the augmentation at the flow $\mathbf{x}$, and critical again in a later augmentation at the flow $\mathbf{y}$. Then $d_{\mathbf{y}}(i) \geq d_{\mathbf{x}}(i) + 2$.

Proof. We apply Lemma 5.13 at every augmentation between $\mathbf{x}$ and $\mathbf{y}$.

1. At $\mathbf{x}$, the arc (i, j) lies on a path with the minimum number of arcs, so $d_{\mathbf{x}}(j) = d_{\mathbf{x}}(i) + 1$; after the augmentation $r_{ij} = 0$.
2. At $\mathbf{y}$ the arc (i, j) is back in the residual network, so some augmentation in between, at a flow $\mathbf{z}$, raised r_{ij} : its path used the reverse arc (j, i) , and $d_{\mathbf{z}}(i) = d_{\mathbf{z}}(j) + 1$.
3. The distances never decrease, so $d_{\mathbf{z}}(j) \geq d_{\mathbf{x}}(j)$ and $d_{\mathbf{y}}(i) \geq d_{\mathbf{z}}(i)$. Hence

$$d_{\mathbf{y}}(i) \geq d_{\mathbf{z}}(i) = d_{\mathbf{z}}(j) + 1 \geq d_{\mathbf{x}}(j) + 1 = d_{\mathbf{x}}(i) + 2.$$

$\square$

Theorem 5.15: running time of Edmonds–Karp

For any non-negative capacities, the Edmonds–Karp algorithm stops after at most nm augmentations with a maximum flow and a minimum cut, and it runs in $O(nm(n+m))$ time, that is $O(nm^2)$ when $m \geq n - 1$.

- Proof.*
1. Every augmentation has at least one critical arc: an arc of P whose residual capacity equals δ .
 2. When (i, j) is critical, i lies before t on a path from s to t with at most $n - 1$ arcs, so $0 \leq d(i) \leq n - 2$. By [Lemma 5.14](#) this distance grows by at least 2 between two critical times, so (i, j) is critical at most $\lfloor \frac{n-2}{2} \rfloor + 1 \leq \frac{n}{2}$ times.
 3. Every residual arc is (i, j) or (j, i) for an arc $(i, j) \in A$: there are at most $2m$ of them, and at most $2m \cdot \frac{n}{2} = nm$ augmentations.
 4. Every iteration takes $O(n + m)$ time, as in [Theorem 5.11](#). At the stop $G(\mathbf{x})$ has no augmenting path, and [Theorem 5.8](#) applies as in the proof of [Theorem 5.9](#). $\square$

- The bound depends only on n and m , not on the capacities: the Edmonds–Karp algorithm is **strongly polynomial**, and it stops also with irrational capacities.

5.6.1 Edmonds–Karp on the reference network

Example 5.8: Edmonds–Karp on the reference network

We run the algorithm on the network of [Example 5.1](#); the column «arcs» is the number of arcs of the augmenting path.

iteration	augmenting path	arcs	r_{ij} on P	backward arcs	δ	v
1	1, 2, 4, 6	3	5, 3, 4	–	3	3
2	1, 3, 5, 6	3	4, 4, 3	–	3	6
3	1, 3, 5, 4, 6	4	1, 1, 2, 1	–	1	7

The distances $d_{\mathbf{x}}(i)$ in the residual network before each search are:

iteration	$d(1)$	$d(2)$	$d(3)$	$d(4)$	$d(5)$	$d(6)$
1	0	1	1	2	2	3
2	0	1	1	3	2	3
3	0	1	1	3	2	4
stop	0	1	∞	∞	∞	∞

- Iteration 1 is the breadth-first search of [Chapter 1, Example «BFS from node 1»](#), whose tree path to node 6 is 1, 2, 4, 6.
- The critical arcs are $(2, 4)$ in iteration 1, $(5, 6)$ in iteration 2, and $(1, 3)$, $(3, 5)$, $(4, 6)$ in iteration 3. When $(2, 4)$ leaves the residual network, $d(4)$ grows from 2 to 3; no distance ever decreases.
- The algorithm needs 3 augmentations, against the 5 of [Example 5.6](#), far below the bound $nm = 54$. It ends with the same maximum flow $\tilde{\mathbf{x}}$, the unique one, and the same cut $S = \{1, 2\}$.

- On the network of [Example 5.7](#) the breadth-first search finds 1, 2, 4 and then 1, 3, 4, each with $\delta = M$: two augmentations for every M .

The Edmonds–Karp algorithm augments along paths with the fewest arcs, found by the breadth-first search. The distances from s in the residual network never decrease, every arc is critical at most $\frac{n}{2}$ times, and the algorithm makes at most nm augmentations: $O(nm^2)$ time, for any capacities.

5.7 Applications

5.7.1 Maximum matching in a bipartite graph

Definition 5.16: bipartite graph and matching

An undirected graph $H = (V_1 \cup V_2, E)$ is **bipartite** if V_1 and V_2 are disjoint and every edge has one endpoint in V_1 and one in V_2 ; we write the edges as $\{i, j\}$ with $i \in V_1$ and $j \in V_2$.

- A **matching** is a set $M \subseteq E$ of edges, no two of which have a common endpoint.
- The **maximum matching problem** asks for a matching with the largest number of edges.

The **network of H** has the nodes $\{s\} \cup V_1 \cup V_2 \cup \{t\}$, the arcs (s, i) for $i \in V_1$, (i, j) for every edge $\{i, j\} \in E$, (j, t) for $j \in V_2$, and all capacities equal to 1.

- With V_1 a set of workers, V_2 a set of jobs, and an edge when a worker can do a job, a matching assigns jobs to workers, at most one job per worker and one worker per job.
- In the order of the nodes, s comes first and t last; the other nodes keep their indices.

Proposition 5.17: matchings and flows

The maximum number of edges of a matching of H equals the maximum value $\tilde{v}$ of a flow in the network of H . If $\tilde{x}$ is an integer maximum flow, then $M = \{\{i, j\} \in E : \tilde{x}_{ij} = 1\}$ is a maximum matching.

Proof. 1. *From matchings to flows.* Given a matching M , set $x = 1$ on the arcs (s, i) , (i, j) , (j, t) of every edge $\{i, j\} \in M$, and $x = 0$ elsewhere. Every node of $V_1 \cup V_2$ is an endpoint of at most one edge of M , so the flow is feasible, and its value is $|M|$.

2. *From integer flows to matchings.* Let x be an integer feasible flow, so $x_{ij} \in \{0, 1\}$. A node $i \in V_1$ has the single entering arc (s, i) , so at most one arc (i, j) carries flow; a node $j \in V_2$ has the single leaving arc (j, t) , so at most one arc (i, j) carries flow. Hence $M = \{\{i, j\} : x_{ij} = 1\}$ is a matching. For $S = \{s\} \cup V_1$, the forward arcs are the arcs (i, j) and there are no backward arcs: by [Lemma 5.3](#),

$$v = |M|.$$

3. By [Corollary 5.10](#) there is an integer maximum flow $\tilde{x}$. By 1, no matching has more than $\tilde{v}$ edges, and by 2 the matching of $\tilde{x}$ has $\tilde{v}$ edges. $\square$

Example 5.9: workers and jobs

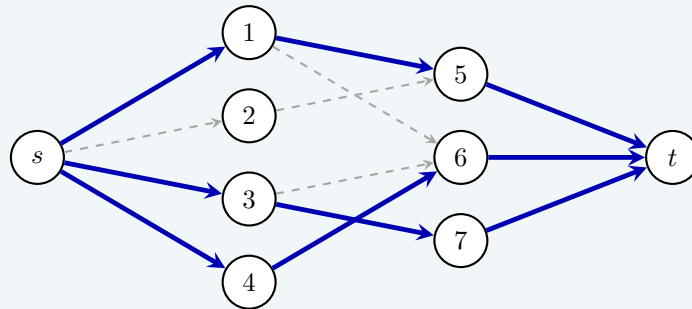
Four workers $V_1 = \{1, 2, 3, 4\}$ and three jobs $V_2 = \{5, 6, 7\}$ have the edges

$$E = \{\{1, 5\}, \{1, 6\}, \{2, 5\}, \{3, 6\}, \{3, 7\}, \{4, 6\}\}.$$

The Edmonds–Karp algorithm on the network of H , with the nodes in the order $s, 1, 2, \dots, 7, t$, gives:

iteration	augmenting path	r_{ij} on P	backward arcs	δ	v
1	$s, 1, 5, t$	1, 1, 1	–	1	1
2	$s, 3, 6, t$	1, 1, 1	–	1	2
3	$s, 4, 6, 3, 7, t$	1, 1, 1, 1, 1	(6, 3)	1	3

- Iteration 3 uses the backward arc (6, 3): worker 3 leaves job 6 for job 7, and worker 4 takes job 6.
- The maximum matching is $M = \{\{1, 5\}, \{3, 7\}, \{4, 6\}\}$, with 3 edges; worker 2 stays without a job. In the figure the arcs with flow 1 are blue.
- At the stop the search labels $S = \{s, 1, 2, 4, 5, 6\}$; the forward arcs of the cut are $(s, 3)$, $(5, t)$, $(6, t)$, with capacity $3 = |M|$.



5.7.2 Arc-disjoint paths

- Two directed paths are **arc-disjoint** if they have no common arc; they may share nodes.

Theorem 5.18: Menger, arc version

Let $G = (N, A)$ be a directed graph and $s \neq t$ two nodes. The maximum number of pairwise arc-disjoint directed paths from s to t equals the minimum number of arcs whose removal leaves no directed path from s to t .

Proof. Give every arc the capacity 1, and let $\tilde{v}$ be the maximum value of a flow.

1. *Paths give flows.* The sum of the incidence vectors of k arc-disjoint paths from s to t is a feasible flow of value k : every arc carries at most 1, and every path enters and leaves its inner nodes. Hence $k \leq \tilde{v}$.
2. *Flows give paths.* By [Corollary 5.10](#) there is a flow x with $x_{ij} \in \{0, 1\}$ and value $\tilde{v}$. If $\tilde{v} \geq 1$, walk from s along arcs with $x_{ij} = 1$, each used at most once, until t is reached:
 - at a node $i \notin \{s, t\}$ the walk has used one entering arc more than leaving arcs, and conservation gives an unused leaving arc with flow;
 - at s the walk has used as many leaving arcs as entering ones, while the arcs with flow leaving s are $\tilde{v} \geq 1$ more than those entering: an unused one exists.

The walk ends, at t , and it contains a path P from s to t ([Chapter 1, Lemma «a walk contains a path»](#)). Setting $x_{ij} = 0$ on P leaves a 0/1 flow of value $\tilde{v} - 1$; repeating gives $\tilde{v}$ arc-disjoint paths.

3. *Removing arcs.* If removing a set F of arcs leaves no path from s to t , the set S of the nodes still reachable from s has $t \notin S$ and $\delta^+(S) \subseteq F$, so $|F| \geq u(S, N \setminus S) \geq \tilde{v}$. Conversely, every path from s to t leaves S through an arc of $\delta^+(S)$: removing the $\tilde{v}$ forward arcs of a minimum cut ([Theorem 5.8](#)) leaves no path. $\square$

Example 5.10: arc-disjoint paths in the reference graph

On the reference graph with all capacities 1, $s = 1$ and $t = 6$, the Edmonds–Karp algorithm finds the paths 1, 2, 4, 6 and 1, 3, 5, 6, and $\tilde{v} = 2$. These two paths are arc-disjoint, and removing the two arcs (1, 2), (1, 3) leaving node 1 leaves no path from 1 to 6: both numbers of [Theorem 5.18](#) are 2.

A maximum matching of a bipartite graph is an integer maximum flow in a network with unit capacities, and so are the arc-disjoint paths from s to t ; the minimum cut certifies both.

5.8 Typical mistakes

Warning

- **Counting the backward arcs of a cut.** The capacity $u(S, N \setminus S)$ adds up only the arcs from S to $N \setminus S$.
- **Forgetting the backward arcs of the residual network.** With forward arcs only, the method can stop at a flow that is not maximum, as at value 5 on the reference network.
- **Giving a backward arc the capacity u_{ij} .** The backward arc (j, i) has $r_{ji} = x_{ij}$: only the flow already sent can be sent back.
- **Taking the bottleneck from the capacities.** The bottleneck is the smallest residual capacity r_{ij} on the path, not the smallest u_{ij} .
- **Adding δ on a backward arc.** Along a backward arc (j, i) the flow x_{ij} decreases by δ .
- **Reading the cut from the saturated arcs.** The minimum cut is given by the nodes reachable from s in the residual network at the stop; the saturated arcs need not form a cut.
- **Calling Ford–Fulkerson polynomial.** Its number of augmentations can grow with the capacities, as $2M$ in [Example 5.7](#); Edmonds–Karp makes at most nm .

5.9 In summary

Summary

- A feasible flow respects the capacities and is conserved at the nodes other than s and t ; its value is the net outflow of s , and it equals the net flow across every s – t cut.
- Weak duality: $v \leq u(S, N \setminus S)$ for every flow and every cut.
- The residual network has forward arcs with $r_{ij} = u_{ij} - x_{ij}$ and backward arcs with $r_{ji} = x_{ij}$; an augmenting path raises the value by its bottleneck δ .
- Max-flow min-cut: a flow is maximum if and only if no augmenting path exists, and $\tilde{v}$ equals the minimum cut capacity; with integer capacities there is an integer maximum flow.
- Ford–Fulkerson (depth-first search): with integer capacities at most $\tilde{v}$ augmentations, pseudopolynomial. Edmonds–Karp (breadth-first search): at most nm augmentations, $O(nm^2)$ time.
- Maximum matchings in bipartite graphs and arc-disjoint paths are integer maximum flows.

The reference network at a glance.

capacities, $s = 1$, $t = 6$	$\mathbf{u} = (5, 4, 3, 2, 4, 2, 4, 2, 3)$
maximum flow (unique)	$\tilde{\mathbf{x}} = (3, 4, 3, 0, 4, 0, 4, 1, 3)$, $\tilde{v} = 7$
minimum cuts, capacity 7	$S = \{1, 2\}, \{1, 2, 3\}, \{1, 2, 3, 4, 5\}$
Ford–Fulkerson (DFS)	5 augmentations, $\delta = 2, 1, 2, 1, 1$
Edmonds–Karp (BFS)	3 augmentations, $\delta = 3, 3, 1$
all capacities 1	arc-disjoint paths 1, 2, 4, 6 and 1, 3, 5, 6

Minimum cost flow

Prerequisites

- From this module, [Chapter 3 «Shortest paths»](#): the Bellman–Ford algorithm with the detection of a negative cycle, Dijkstra’s algorithm, and the potentials $\pi_i = -\tilde{d}(i)$ as a certificate.
- [Chapter 4 «Network flow models»](#): the minimum cost flow model, the flow across a cut, the substitution of the lower bounds, the potentials and the reduced costs.
- [Chapter 5 «Maximum flow»](#): the residual network $G(\mathbf{x})$, augmenting paths and the Edmonds–Karp algorithm.

Objectives

At the end of this chapter you will be able to:

- build the residual network of a flow with residual capacities and costs, compute the cost of one of its cycles, and augment the flow along it;
- decompose a circulation, and the difference of two feasible flows, into cycles;
- prove that a feasible flow is optimal if and only if its residual network has no negative cycle, and if and only if potentials with non-negative reduced costs exist, and recognize an unbounded instance;
- find a first feasible flow with a maximum flow;
- run the cycle-canceling algorithm and the successive shortest path algorithm, and prove their correctness.

Where we were

- In [Chapter 4 «Network flow models»](#) the minimum cost flow problem is the LP $\min\{\mathbf{c}'\mathbf{x} : \mathbf{B}\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$. Potentials π_i whose reduced costs $\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j$ are positive only on empty arcs and negative only on saturated arcs certify that a flow is optimal.

- On the reference network the optimal flow $\tilde{\mathbf{x}}$ costs 60, and $\boldsymbol{\pi} = (15, 13, 13, 5, 10, 0)$ certifies it.
- In [Chapter 5 «Maximum flow»](#) a flow grows along augmenting paths of the residual network $G(\mathbf{x})$.
- In this chapter the certificate becomes an algorithm: we lower the cost of a feasible flow along cycles of $G(\mathbf{x})$, or we build the flow along shortest paths of $G(\mathbf{x})$ keeping the certificate true.

6.1 The residual network with costs

- In this chapter the network $G = (N, A)$ has balances b_i with $\sum_{i \in N} b_i = 0$, lower bounds l_{ij} , capacities u_{ij} (finite or $+\infty$) and costs c_{ij} , as in [Chapter 4](#). The arcs are numbered $a_1, a_2, \dots, a_m$ in lexicographic order.
- The set A has no pair of opposite arcs $(i, j), (j, i)$. An arc (j, i) opposite to (i, j) can be replaced by the arcs (j, k) and (k, i) through a new transshipment node k , both with the bounds of (j, i) , the first with its cost and the second with cost 0: the flows and their costs do not change.

Definition 6.1: residual network

Let $\mathbf{x}$ be a vector with $l_{ij} \leq x_{ij} \leq u_{ij}$ for every arc. The **residual network** $G(\mathbf{x}) = (N, A(\mathbf{x}))$ has, for every arc $(i, j) \in A$:

- the **forward arc** (i, j) , if $x_{ij} < u_{ij}$, with **residual capacity** $r_{ij} = u_{ij} - x_{ij}$ and cost c_{ij} ;
- the **backward arc** (j, i) , if $x_{ij} > l_{ij}$, with residual capacity $r_{ji} = x_{ij} - l_{ij}$ and cost $-c_{ij}$.

We write c_{ij}^x for the cost of the arc (i, j) of $G(\mathbf{x})$.

- A forward arc can carry r_{ij} more units on (i, j) . A backward arc (j, i) sends units back from j to i by decreasing x_{ij} , and saves c_{ij} per unit.
- An arc of A gives two arcs of $G(\mathbf{x})$ exactly when $l_{ij} < x_{ij} < u_{ij}$. With $\mathbf{l} = \mathbf{0}$ and without the costs, $G(\mathbf{x})$ is the residual network of [Chapter 5 «Maximum flow»](#).

Definition 6.2: cycles of the residual network

Let W be a directed cycle of $G(\mathbf{x})$. Its **cost** and its **residual capacity** are

$$c(W) = \sum_{(i,j) \in W} c_{ij}^x, \quad r(W) = \min_{(i,j) \in W} r_{ij},$$

and W is a **negative cycle** if $c(W) < 0$. **Augmenting** $\mathbf{x}$ along W by θ , with $0 < \theta \leq r(W)$, means increasing x_{ij} by θ for every forward arc (i, j) of W , and decreasing x_{ij} by θ for every arc $(i, j) \in A$ whose backward arc (j, i) is in W .

Proposition 6.3: augmenting along a cycle

Let $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$, let W be a cycle of $G(\mathbf{x})$ and $0 < \theta \leq r(W)$, and let $\mathbf{x}'$ be $\mathbf{x}$ augmented along W by θ . Then $\mathbf{l} \leq \mathbf{x}' \leq \mathbf{u}$, every node has the same outflow minus inflow in $\mathbf{x}'$ and in $\mathbf{x}$, and

$$\mathbf{c}'\mathbf{x}' = \mathbf{c}'\mathbf{x} + \theta c(W).$$

Proof. We check the three claims arc by arc.

1. *Bounds.* On a forward arc, $x'_{ij} = x_{ij} + \theta \leq x_{ij} + r_{ij} = u_{ij}$. If the backward arc (j, i) is in W , then $x'_{ij} = x_{ij} - \theta \geq x_{ij} - r_{ji} = l_{ij}$.
2. *Balances.* Augmenting by θ along an arc (i, k) of $G(\mathbf{x})$ raises the outflow minus inflow of i by θ and lowers that of k by θ : a forward arc sends more flow from i to k , a backward arc sends less flow from k to i . Every node of W is the tail of one arc of W and the head of another one, so the changes cancel.
3. *Cost.* A forward arc changes the cost by $c_{ij}\theta = c_{ij}^x\theta$; a backward arc (j, i) changes it by $-c_{ij}\theta = c_{ji}^x\theta$. Adding up along W gives $\theta c(W)$. $\square$

- If $\mathbf{x}$ is a feasible flow, so is $\mathbf{x}'$. Along a negative cycle the cost decreases by $\theta |c(W)|$.

Example 6.1: the residual network of the most expensive flow

We use the reference network of [Chapter 4](#), with zero lower bounds, the balances $\mathbf{b} = (5, 2, 0, -3, 0, -4)$, and the costs and capacities

arc	(1, 2)	(1, 3)	(2, 4)	(3, 2)	(3, 5)	(4, 3)	(4, 6)	(5, 4)	(5, 6)
c_{ij}	2	2	5	2	3	3	5	5	3
u_{ij}	4	4	4	2	4	3	5	3	2

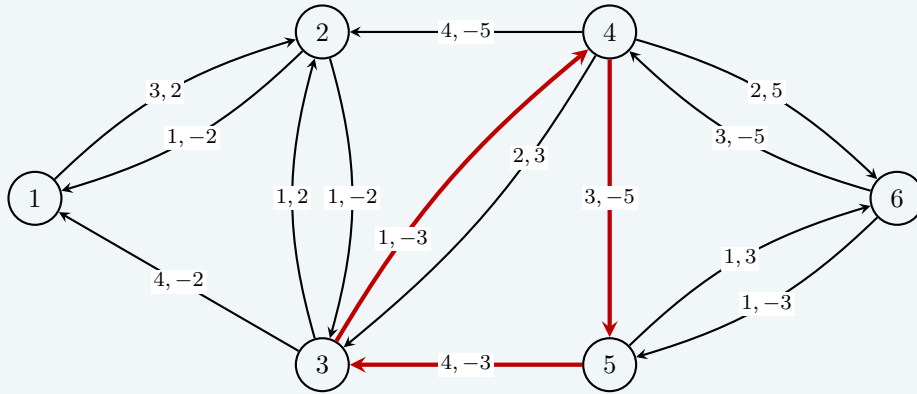
Its most expensive feasible flow, computed with Gurobi by maximizing the cost, is

$$\mathbf{x}^0 = (1, 4, 4, 1, 4, 1, 3, 3, 1), \quad \mathbf{c}'\mathbf{x}^0 = 80,$$

with the entries in the order of the arcs; it is the only flow of cost 80, as the enumeration of the 12 integer feasible flows confirms. Each arc gives the following arcs of $G(\mathbf{x}^0)$, with residual capacity and cost:

arc (i, j)	(1, 2)	(1, 3)	(2, 4)	(3, 2)	(3, 5)	(4, 3)	(4, 6)	(5, 4)	(5, 6)
x_{ij}^0	1	4	4	1	4	1	3	3	1
forward (i, j) : r, c	3, 2	—	—	1, 2	—	2, 3	2, 5	—	1, 3
backward (j, i) : r, c	1, -2	4, -2	4, -5	1, -2	4, -3	1, -3	3, -5	3, -5	1, -3

In the figure each arc of $G(\mathbf{x}^0)$ carries the pair r_{ij}, c_{ij}^x ; the cycle $W = 4, 5, 3, 4$ is red.



- The cycle $W = 4, 5, 3, 4$ uses the backward arcs of $(5, 4)$, $(3, 5)$ and $(4, 3)$:

$$c(W) = -5 - 3 - 3 = -11, \quad r(W) = \min\{3, 4, 1\} = 1.$$

- Augmenting by $\theta = 1$ decreases x_{54} , x_{35} and x_{43} by one unit. The new flow $\mathbf{x}^1 = (1, 4, 4, 1, 3, 0, 3, 2, 1)$ is feasible and costs $80 - 11 = 69$: the unit that went around the directed cycle $3, 5, 4, 3$ of G cost 11 and delivered nothing.

The residual network $G(\mathbf{x})$ has a forward arc of cost c_{ij} where x_{ij} can grow and a backward arc of cost $-c_{ij}$ where x_{ij} can decrease. Augmenting along a cycle of $G(\mathbf{x})$ keeps the bounds and the balances, and changes the cost by $\theta c(W)$.

6.2 Circulations and their decomposition

Definition 6.4: circulation

Let $H = (N, F)$ be a directed graph. A **circulation** on H is a vector $\mathbf{y} \in \mathbb{R}^F$ with $\mathbf{y} \geq 0$ such that every node has outflow equal to inflow:

$$\sum_{(i,j) \in \delta^+(i)} y_{ij} = \sum_{(j,i) \in \delta^-(i)} y_{ji}, \quad \forall i \in N.$$

The **incidence vector** χ^W of a directed cycle W of H , with $\chi_a^W = 1$ on the arcs of W and $\chi_a^W = 0$ elsewhere, is a circulation.

Lemma 6.5: decomposition into cycles

Let $\mathbf{y} \neq \mathbf{0}$ be a circulation on $H = (N, F)$, with p arcs where $y_a > 0$. Then there are directed cycles $W_1, W_2, \dots, W_k$ of H , with $k \leq p$, made of arcs with $y_a > 0$, and numbers $\theta_1, \theta_2, \dots, \theta_k > 0$ such that

$$\mathbf{y} = \sum_{h=1}^k \theta_h \chi^{W_h}.$$

If $\mathbf{y}$ is integer, the numbers θ_h are integers.

Proof. By induction on p .

1. *A cycle.* Let (i_0, i_1) be an arc with $y_{i_0 i_1} > 0$. Node i_1 receives a positive inflow, so by the balance some arc (i_1, i_2) has $y_{i_1 i_2} > 0$. Going on in the same way we build a directed walk of arcs with $y > 0$. Since N is finite a node repeats, and the piece of the walk between its two occurrences is a directed cycle W_1 .
2. *Removing it.* Let $\theta_1 = \min_{a \in W_1} y_a > 0$. The vector $\mathbf{y}' = \mathbf{y} - \theta_1 \chi^{W_1}$ is non-negative, it is balanced as a difference of balanced vectors, and it has at most $p - 1$ positive arcs, since $y'_a = 0$ on an arc of W_1 that attains the minimum.
3. *Induction.* If $\mathbf{y}' = \mathbf{0}$, then $\mathbf{y} = \theta_1 \chi^{W_1}$. Otherwise the induction gives $\mathbf{y}' = \sum_{h=2}^k \theta_h \chi^{W_h}$ with $k - 1 \leq p - 1$ cycles of arcs with $y'_a > 0$, hence $y_a > 0$.

If $\mathbf{y}$ is integer, θ_1 is one of its entries and $\mathbf{y}'$ is integer, so all the θ_h are integers. $\square$

Lemma 6.6: the difference of two feasible flows

Let $\mathbf{x}$ and $\mathbf{x}'$ be feasible flows. Define $\mathbf{y}$ on the arcs of $G(\mathbf{x})$: for every $(i, j) \in A$,

$$\begin{aligned} y_{ij} &= x'_{ij} - x_{ij} && \text{on the forward arc } (i, j), \text{ if } x'_{ij} > x_{ij}, \\ y_{ji} &= x_{ij} - x'_{ij} && \text{on the backward arc } (j, i), \text{ if } x'_{ij} < x_{ij}, \end{aligned}$$

and $y = 0$ on the other arcs of $G(\mathbf{x})$. Then $\mathbf{y}$ is a circulation on $G(\mathbf{x})$ with $y_{ij} \leq r_{ij}$, and

$$\mathbf{c}'\mathbf{x}' - \mathbf{c}'\mathbf{x} = \sum_{(i,j) \in A(\mathbf{x})} c_{ij}^x y_{ij} = \sum_{h=1}^k \theta_h c(W_h)$$

for every decomposition $\mathbf{y} = \sum_{h=1}^k \theta_h \chi^{W_h}$ into cycles of $G(\mathbf{x})$.

Proof. We check that $\mathbf{y}$ lives on $G(\mathbf{x})$, that it is balanced, and its cost.

1. If $x'_{ij} > x_{ij}$, then $x_{ij} < x'_{ij} \leq u_{ij}$: the forward arc exists and $y_{ij} \leq u_{ij} - x_{ij} = r_{ij}$.
If $x'_{ij} < x_{ij}$, then $x_{ij} > x'_{ij} \geq l_{ij}$: the backward arc exists and $y_{ji} \leq x_{ij} - l_{ij} = r_{ji}$.
2. As in the proof of Proposition 6.3, y on an arc (i, k) of $G(\mathbf{x})$ adds y to the outflow minus inflow of i and subtracts it from that of k , exactly as the change from $\mathbf{x}$ to $\mathbf{x}'$ does on the corresponding arc of A . Hence the outflow minus inflow of $\mathbf{y}$ at node i is $(B\mathbf{x}')_i - (B\mathbf{x})_i = b_i - b_i = 0$.
3. A forward arc contributes $c_{ij}(x'_{ij} - x_{ij})$, a backward arc $-c_{ij}(x_{ij} - x'_{ij}) = c_{ij}(x'_{ij} - x_{ij})$: the total is $\mathbf{c}'(\mathbf{x}' - \mathbf{x})$. With the decomposition, $\sum c_{ij}^x y_{ij} = \sum_h \theta_h \sum_{(i,j) \in W_h} c_{ij}^x = \sum_h \theta_h c(W_h)$. $\square$

- By Lemma 6.5 such a decomposition exists: $\mathbf{x}'$ is $\mathbf{x}$ augmented along the cycles $W_1, W_2, \dots, W_k$ of $G(\mathbf{x})$ by $\theta_1, \theta_2, \dots, \theta_k$. Since $\mathbf{y}$ is positive on at most one arc of $G(\mathbf{x})$ per arc of A , $k \leq m$.

Example 6.2: from the most expensive flow to the optimal flow

We compare $\mathbf{x}^0$ of [Example 6.1](#), of cost 80, with the optimal flow $\tilde{\mathbf{x}} = (2, 3, 4, 0, 3, 0, 2, 1, 2)$ of Chapter 4, of cost 60. The difference is

$$\tilde{\mathbf{x}} - \mathbf{x}^0 = (1, -1, 0, -1, -1, -1, -2, 1),$$

so $\mathbf{y}$ is positive on eight arcs of $G(\mathbf{x}^0)$: the forward arcs $(1, 2)$ and $(5, 6)$, with $y = 1$, and the backward arcs

$$y_{31} = 1, \quad y_{23} = 1, \quad y_{53} = 1, \quad y_{34} = 1, \quad y_{64} = 1, \quad y_{45} = 2.$$

We follow the proof of [Lemma 6.5](#), starting each time from the first arc with $y > 0$ in lexicographic order and leaving every node along its arc with $y > 0$ of smallest head:

h	cycle W_h	θ_h	$c(W_h)$
1	1, 2, 3, 1	1	$2 - 2 - 2 = -2$
2	3, 4, 5, 3	1	$-3 - 5 - 3 = -11$
3	4, 5, 6, 4	1	$-5 + 3 - 5 = -7$

By [Lemma 6.6](#), $\mathbf{c}'\tilde{\mathbf{x}} - \mathbf{c}'\mathbf{x}^0 = -2 - 11 - 7 = -20 = 60 - 80$. All three cycles are negative, and W_2 is the cycle 4, 5, 3, 4 of [Example 6.1](#), written from node 3.

A circulation is a sum of positive multiples of directed cycles. The difference of two feasible flows is a circulation on $G(\mathbf{x})$: every feasible flow is reached from $\mathbf{x}$ along cycles of $G(\mathbf{x})$, and the cost changes by $\sum_h \theta_h c(W_h)$.

6.3 Negative cycle optimality conditions

Theorem 6.7: negative cycle optimality

A feasible flow $\mathbf{x}$ is optimal if and only if its residual network $G(\mathbf{x})$ has no negative cycle.

Proof. We prove the two implications.

- *A negative cycle improves $\mathbf{x}$.* Let W be a negative cycle of $G(\mathbf{x})$ and $\theta = \min\{r(W), 1\} > 0$. By [Proposition 6.3](#), $\mathbf{x}$ augmented along W by θ is a feasible flow of cost $\mathbf{c}'\mathbf{x} + \theta c(W) < \mathbf{c}'\mathbf{x}$, so $\mathbf{x}$ is not optimal.
- *Without negative cycles $\mathbf{x}$ is optimal.* Let $\mathbf{x}'$ be any feasible flow. By [Lemma 6.6](#), $\mathbf{c}'\mathbf{x}' = \mathbf{c}'\mathbf{x} + \sum_h \theta_h c(W_h)$ with cycles W_h of $G(\mathbf{x})$ and $\theta_h > 0$. Every $c(W_h)$ is non-negative, so $\mathbf{c}'\mathbf{x}' \geq \mathbf{c}'\mathbf{x}$. $\square$

- The proof compares two flows through their difference, and uses no LP duality.
- When some capacities are infinite the model can be unbounded, and the next proposition tells exactly when.

Proposition 6.8: unbounded instances

Assume that the MCF model has a feasible flow. It is unbounded if and only if G has a directed cycle C of negative cost, $\sum_{(i,j) \in C} c_{ij} < 0$, whose arcs all have $u_{ij} = +\infty$.

Proof. If. Let $\mathbf{x}$ be feasible. Every arc of C has $x_{ij} < u_{ij} = +\infty$, so C is a cycle of forward arcs of $G(\mathbf{x})$ with $r(C) = +\infty$. Augmenting along C by any $\theta > 0$ gives feasible flows of cost $\mathbf{c}'\mathbf{x} + \theta c(C)$, which tends to $-\infty$.

Only if. Assume that there is no such cycle C , and fix a feasible flow $\mathbf{x}$. Let R be the largest finite residual capacity of an arc of $G(\mathbf{x})$ ($R = 0$ if there is none), and $K = \max_{(i,j) \in A} |c_{ij}|$. Let $\mathbf{x}'$ be any feasible flow, decomposed as in Lemma 6.6.

1. A cycle W_h with $c(W_h) < 0$ has an arc a of finite residual capacity: backward arcs have finite residual capacity, and a cycle of forward arcs with infinite capacity would be a cycle C . Hence $\theta_h \leq y_a \leq r_a \leq R$.
2. W_h has at most n arcs, so $c(W_h) \geq -nK$; there are at most m cycles.

Hence $\mathbf{c}'\mathbf{x}' \geq \mathbf{c}'\mathbf{x} - m R n K$ for every feasible $\mathbf{x}'$: the model is bounded. $\square$

- A feasible LP that is bounded has an optimal solution (Linear Programming, Chapter 15 «The geometry of Linear Programming»). With finite capacities the MCF model is never unbounded.

Example 6.3: a flow that is not optimal and an unbounded instance

- The feasible flow $\bar{\mathbf{x}} = (1, 4, 3, 0, 4, 0, 2, 2, 2)$ of Chapter 4 costs 63. Its residual network has the cycle $W = 1, 2, 4, 5, 3, 1$, made of the forward arcs $(1, 2)$, $(2, 4)$ and of the backward arcs $(4, 5)$, $(5, 3)$, $(3, 1)$:

$$c(W) = 2 + 5 - 5 - 3 - 2 = -3, \quad r(W) = \min\{3, 1, 2, 4, 4\} = 1.$$

By Theorem 6.7 $\bar{\mathbf{x}}$ is not optimal. Augmenting by 1 gives $\tilde{\mathbf{x}}$, of cost $63 - 3 = 60$: the gap 3 of $\bar{\mathbf{x}}$ computed with the dual solution in Chapter 4.

- With $c_{54} = -12$ and $u_{35} = u_{54} = u_{43} = +\infty$, the directed cycle $3, 5, 4, 3$ of G costs $3 - 12 + 3 = -6$ and has infinite capacities. By Proposition 6.8 the model is unbounded, as Gurobi confirms: each unit sent around the cycle lowers the cost by 6.

A feasible flow is optimal if and only if its residual network has no negative cycle. A feasible model is unbounded if and only if a negative directed cycle has all its capacities infinite.

6.4 Optimality conditions with potentials

- Given potentials $\pi \in \mathbb{R}^n$, the reduced cost of an arc (i, j) of $G(\mathbf{x})$ is $c_{ij}^x - \pi_i + \pi_j$. For a forward arc it is $\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j$, the reduced cost of Chapter 4; for the backward arc (j, i) of (i, j) it is $-c_{ij} - \pi_j + \pi_i = -\bar{c}_{ij}$.
- Along a cycle W of $G(\mathbf{x})$ the potentials telescope, as in Chapter 3:

$$c(W) = \sum_{(i,j) \in W} (c_{ij}^x - \pi_i + \pi_j).$$

Theorem 6.9: reduced cost optimality

A feasible flow $\mathbf{x}$ is optimal if and only if there are potentials $\pi \in \mathbb{R}^n$ such that

$$c_{ij}^x - \pi_i + \pi_j \geq 0 \quad \text{for every arc } (i, j) \text{ of } G(\mathbf{x}).$$

Proof. If. By the telescoping formula every cycle of $G(\mathbf{x})$ has $c(W) \geq 0$, and $\mathbf{x}$ is optimal by Theorem 6.7.

Only if. Let $\mathbf{x}$ be optimal: by Theorem 6.7, $G(\mathbf{x})$ has no negative cycle.

1. Add a node r and an arc (r, i) of cost 0 for every $i \in N$. In the new graph every node is reachable from r , and the cycles are those of $G(\mathbf{x})$, since no arc enters r : none is negative.
2. By Theorem «Bellman's equations» of Chapter 3, the distances $d(i)$ from r are finite and $d(j) \leq d(i) + c_{ij}^x$ for every arc (i, j) of $G(\mathbf{x})$.
3. With $\pi_i = -d(i)$ we get $c_{ij}^x - \pi_i + \pi_j = c_{ij}^x + d(i) - d(j) \geq 0$. □

Corollary 6.10: complementary slackness

Let $\mathbf{x}$ be a feasible flow and $\pi \in \mathbb{R}^n$. The condition of Theorem 6.9 holds if and only if, for every arc $(i, j) \in A$,

$$\bar{c}_{ij} > 0 \Rightarrow x_{ij} = l_{ij}, \quad \bar{c}_{ij} < 0 \Rightarrow x_{ij} = u_{ij}.$$

In particular, $l_{ij} < x_{ij} < u_{ij}$ implies $\bar{c}_{ij} = 0$.

Proof. The arc $(i, j) \in A$ gives the forward arc, of reduced cost $\bar{c}_{ij}$, when $x_{ij} < u_{ij}$, and the backward arc, of reduced cost $-\bar{c}_{ij}$, when $x_{ij} > l_{ij}$. So the condition on the arcs coming from (i, j) reads

$$x_{ij} < u_{ij} \Rightarrow \bar{c}_{ij} \geq 0, \quad x_{ij} > l_{ij} \Rightarrow \bar{c}_{ij} \leq 0.$$

Since $l_{ij} \leq x_{ij} \leq u_{ij}$, the first implication is equivalent to $\bar{c}_{ij} < 0 \Rightarrow x_{ij} = u_{ij}$, and the second to $\bar{c}_{ij} > 0 \Rightarrow x_{ij} = l_{ij}$. If $l_{ij} < x_{ij} < u_{ij}$, both implications apply and $\bar{c}_{ij} = 0$. □

- With $l = 0$ these are the conditions of Corollary «reduced cost certificate» of Chapter 4. There they were proved sufficient with LP duality; Theorem 6.9 shows that every optimal flow has such potentials, which with $\alpha_{ij} = \max\{0, -\bar{c}_{ij}\}$ form an optimal dual solution.
- The potentials of the proof are $\pi_i = -d(i)$, minus the distances from r , as $\pi_i = -\tilde{d}(i)$ in Chapter 3. They can be shifted by a constant; we normalize them with $\pi_n = 0$.

Example 6.4: the certificate of the optimal flow

We take $\tilde{x}$ and the potentials $\pi = (15, 13, 13, 5, 10, 0)$ of Chapter 4. The reduced costs of the arcs of $G(\tilde{x})$ are $\bar{c}_{ij}$ on the forward arcs and $-\bar{c}_{ij}$ on the backward arcs:

arc (i, j)	(1, 2)	(1, 3)	(2, 4)	(3, 2)	(3, 5)	(4, 3)	(4, 6)	(5, 4)	(5, 6)
$\tilde{x}_{ij}$	2	3	4	0	3	0	2	1	2
$\bar{c}_{ij}$	0	0	-3	2	0	11	0	0	-7
forward (i, j)	0	0	-	2	0	11	0	0	-
backward (j, i)	0	0	3	-	0	-	0	0	7

- All the 14 arcs of $G(\tilde{x})$ have non-negative reduced cost: by Theorem 6.9, $\tilde{x}$ is optimal. In the form of Corollary 6.10: (3, 2) and (4, 3) are empty, (2, 4) and (5, 6) are saturated.
- The potentials are minus the costs of the cheapest walks of $G(\tilde{x})$ ending at each node. For example, the walk 6, 4, 5, 3, 1 uses four backward arcs and costs $-5 - 5 - 3 - 2 = -15 = -\pi_1$: in G it is the path 1, 3, 5, 4, 6 of the spanning tree of Chapter 4.

A feasible flow is optimal if and only if some potentials give non-negative reduced costs on all the arcs of $G(x)$: positive $\bar{c}_{ij}$ only on arcs at the lower bound, negative only on saturated arcs. Such potentials are minus the distances in $G(x)$ from a node joined to all the others.

6.5 A first feasible flow

- An algorithm that improves a feasible flow needs a feasible flow to start from. By the substitution of Chapter 4 we can assume $l = 0$.

Definition 6.11: auxiliary network

Let $G = (N, A)$ be a balanced network with $l = 0$, and $D = \sum_{i \in N: b_i > 0} b_i$ its total supply. The **auxiliary network** adds to G a **super source** s and a **super sink** t , an arc (s, i) with capacity b_i for every supply node i , and an arc (i, t) with capacity $-b_i$ for every demand node i ; the arcs of A keep their capacities.

Proposition 6.12: a feasible flow from a maximum flow

The network G has a feasible flow if and only if the maximum s - t flow of the auxiliary network has value D . In that case, a maximum flow restricted to the arcs of A is a feasible flow of G .

Proof. The value of an s - t flow is the flow leaving s , at most the total capacity D of the arcs (s, i) .

1. *Only if.* Let $\mathbf{x}$ be feasible, and set $x_{si} = b_i$ on the arcs (s, i) and $x_{it} = -b_i$ on the arcs (i, t) . At a supply node the outflow minus inflow becomes $b_i - x_{si} = 0$, at a demand node $b_i + x_{it} = 0$, and transshipment nodes do not change: this is an s - t flow of value D , hence maximum.
2. *If.* Let a maximum flow have value D . Then all the arcs (s, i) are saturated. The value is also the flow entering t , and the arcs (i, t) have total capacity $\sum_{i: b_i < 0} (-b_i) = D$, since the network is balanced: they are saturated too.
3. Conservation at $i \in N$ then gives, along the arcs of A , outflow minus inflow equal to $x_{si} - x_{it} = b_i$, and the capacities hold: the restriction to A is a feasible flow. $\square$

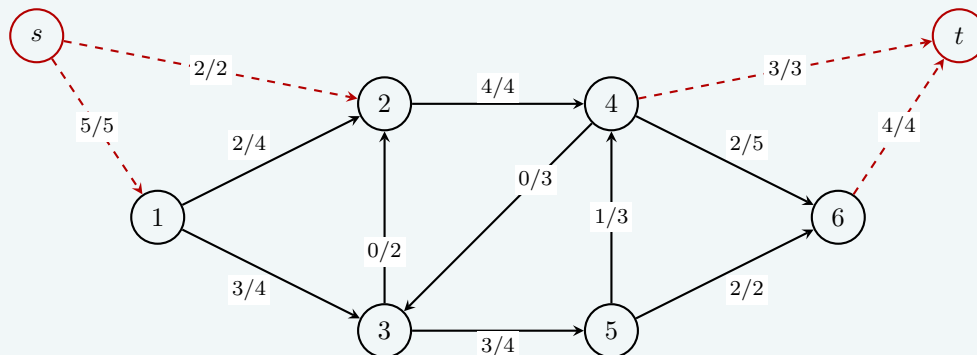
- We compute the maximum flow with the Edmonds–Karp algorithm of Chapter 5: a breadth-first search from s in the residual network, scanning the arcs that leave a node by increasing index of the head (s before node 1, t after node n).
- The maximum flow ignores the costs: it gives a feasible flow, usually far from optimal.

Example 6.5: a feasible flow of the reference network

The reference network has $D = 5 + 2 = 7$. The auxiliary network adds the arcs $(s, 1)$, $(s, 2)$ with capacities 5, 2, and $(4, t)$, $(6, t)$ with capacities 3, 4 (red and dashed in the figure). The augmenting paths of the Edmonds–Karp algorithm are:

path	nodes	units
1	$s, 2, 4, t$	2
2	$s, 1, 2, 4, t$	1
3	$s, 1, 2, 4, 6, t$	1
4	$s, 1, 3, 5, 6, t$	2
5	$s, 1, 3, 5, 4, 6, t$	1

The value is $7 = D$, so the network has a feasible flow. Each arc carries x_{ij}/u_{ij} :



The restriction to A is $(2, 3, 4, 0, 3, 0, 2, 1, 2)$: on this instance it is the optimal flow $\tilde{\mathbf{x}}$ of Chapter 4. This is a coincidence, since the search never looks at the costs.

A super source feeding the supply nodes and a super sink fed by the demand nodes turn feasibility into a maximum flow problem: the network has a feasible flow if and only if the maximum flow saturates all the new arcs.

6.6 The cycle-canceling algorithm

- **Theorem 6.7** suggests an algorithm due to Klein: while $G(\mathbf{x})$ has a negative cycle, augment along it. Augmenting by $r(W)$ saturates an arc of W , which leaves $G(\mathbf{x})$: the cycle is canceled.

6.6.1 Finding a negative cycle

- A negative cycle of $G(\mathbf{x})$ need not be reachable from a given node. We run the Bellman–Ford algorithm of [Chapter 3](#) from a new node r joined to every node by an arc of cost 0; its first pass gives label 0 to every node, and we write the table from there.

Proposition 6.13: Bellman–Ford on the residual network

Let $d^0(j) = 0$ for every $j \in N$ and, for $k \geq 1$,

$$d^k(j) = \min \left\{ d^{k-1}(j), \min_{(i,j) \in A(\mathbf{x})} (d^{k-1}(i) + c_{ij}^x) \right\}, \quad \forall j \in N.$$

- $d^k(j)$ is the minimum cost of a walk of $G(\mathbf{x})$ with at most k arcs that ends at j , the walk without arcs having cost 0.
- $G(\mathbf{x})$ has a negative cycle if and only if $d^n \neq d^{n-1}$. In that case, if $d^n(j) < d^{n-1}(j)$, the predecessors read backwards from j , one row per pass from row n to row 1, give a walk of $G(\mathbf{x})$ with n arcs, ending at j , that contains a negative cycle.
- If $d^k = d^{k-1}$ for some $k \leq n$, the potentials $\pi_i = -d^k(i)$ satisfy $c_{ij}^x - \pi_i + \pi_j \geq 0$ on every arc of $G(\mathbf{x})$.

Proof. Let H be $G(\mathbf{x})$ plus the node r and the arcs (r, i) of cost 0, and let D^k be the labels of Bellman–Ford from r in H : $D^1(i) = 0$ for every $i \in N$, and $D^k(r) = 0$ for every k .

- By induction on k , $D^{k+1}(j) = d^k(j)$ for $j \in N$: the recursions coincide, and the arc (r, j) offers $D^k(r) + 0 = 0$, which never lowers the minimum since $d^{k-1}(j) \leq d^0(j) = 0$. A walk from r with $k + 1$ arcs is an arc (r, i) followed by a walk of $G(\mathbf{x})$ with k arcs from i : this proves (a).
- In H every node is reachable from r , and the cycles are those of $G(\mathbf{x})$. By Theorem «detection of a negative cycle» of Chapter 3, applied to the $n + 1$ nodes of H , a negative cycle exists if and only if $D^{n+1} \neq D^n$, that is $d^n \neq d^{n-1}$. The walk from r with $n + 1$ arcs read in rows $n + 1, \dots, 2$ of H , which are our rows $n, \dots, 1$, contains a negative cycle; without its first arc $(r, \cdot)$ it is a walk of $G(\mathbf{x})$ with n arcs, and the cycle does not contain r . This proves (b).

3. If $d^k = d^{k-1}$, the next pass repeats the labels: $d^k(j) \leq d^k(i) + c_{ij}^x$ for every arc (i, j) of $G(\mathbf{x})$, which is (c) with $\pi = -d^k$. $\square$

- **Rules of this chapter.** The arcs of $G(\mathbf{x})$ are scanned in lexicographic order; as in Chapter 3, ties in the minimum go to the smallest tail and a predecessor changes only when the label strictly decreases. If $d^n \neq d^{n-1}$, j is the smallest node with $d^n(j) < d^{n-1}(j)$, and the cycle is the piece of the walk between the two occurrences of the first node met twice, reading the walk from its first node.
- At the stop the potentials $\pi_i = -d^k(i)$ are shifted so that $\pi_n = 0$.

6.6.2 The algorithm

Algorithm

Algorithm: Cycle-canceling algorithm

Input: a balanced network with $\mathbf{l} = \mathbf{0}$, capacities u_{ij} , costs c_{ij}

Output: an optimal flow $\mathbf{x}$ with its certificate $\boldsymbol{\pi}$, or «infeasible», or «unbounded»

- 1 **Step 0 (start).** If a cycle of arcs with $u_{ij} = +\infty$ has negative cost, STOP: unbounded.
- 2 Find a feasible flow $\mathbf{x}$ with a maximum flow of the auxiliary network.
- 3 If its value is less than D , STOP: infeasible.
- 4 **Step 1 (negative cycle).** Run Bellman–Ford on $G(\mathbf{x})$ from $d^0 = \mathbf{0}$.
- 5 If $d^k = d^{k-1}$ for some $k \leq n$, STOP: $\mathbf{x}$ is optimal, with $\boldsymbol{\pi} = -d^k$.
- 6 Otherwise read a negative cycle W from the table.
- 7 **Step 2 (cancel).** Augment $\mathbf{x}$ along W by $\theta = r(W)$ and go to Step 1.

- The test of Step 0 is [Proposition 6.13](#) applied to the graph of the arcs with $u_{ij} = +\infty$ and their costs. When it fails, every negative cycle of $G(\mathbf{x})$ has an arc of finite residual capacity (proof of [Proposition 6.8](#)), so $r(W) < +\infty$ in Step 2.
- Any feasible flow can be used in Step 0; the maximum flow is one way to find it.

Theorem 6.14: correctness of cycle canceling

Let the balances, the capacities and the costs be integer, with capacities possibly $+\infty$, and let the algorithm pass Step 0 with an integer feasible flow $\mathbf{x}^0$. Then every flow of the algorithm is integer and feasible, every iteration lowers the cost by at least 1, and the algorithm stops after at most $\mathbf{c}'\mathbf{x}^0 - z(\text{MCF})$ iterations with an optimal flow.

Proof. By [Proposition 6.8](#) the model is bounded, so it has an optimal value $z(\text{MCF})$.

1. If $\mathbf{x}$ is integer, the residual capacities are positive integers or $+\infty$, and $r(W)$ is a positive integer, since $r(W) < +\infty$. By [Proposition 6.3](#) the new flow is feasible and integer.

2. The cost changes by $r(W)c(W)$, where $r(W) \geq 1$ and $c(W) \leq -1$ are integers: it decreases by at least 1.
3. The cost starts at $c'x^0$ and never goes below $z(\text{MCF})$, so there are at most $c'x^0 - z(\text{MCF})$ iterations. At the stop $G(x)$ has no negative cycle ([Proposition 6.13](#)), and x is optimal by [Theorem 6.7](#). $\square$

- **Running time.** $G(x)$ has at most $2m$ arcs, so Step 1 takes $O(nm)$ time. With finite capacities, $U = \max u_{ij}$ and $K = \max |c_{ij}|$, every flow costs between $-mKU$ and mKU : at most $2mKU$ iterations, and $O(nm^2KU)$ time.
- The bound depends on the size of the numbers K and U , not only on n and m : it is a **pseudo-polynomial** bound.

6.6.3 Cycle canceling on the reference network

- On the reference network Step 0 returns $\tilde{x}$ ([Example 6.5](#)), and the algorithm stops at the first Bellman–Ford run. To see it at work, we start from the most expensive flow x^0 of [Example 6.1](#), of cost 80.
- In the tables each row is a pass k ; next to a label that decreased, its new predecessor in brackets. The entries read backwards to find the walk are in bold.

Example 6.6: cycle canceling from the most expensive flow

Iteration 1

$G(x^0)$ is the residual network of [Example 6.1](#). Bellman–Ford gives:

k	$d^k(1)$	$d^k(2)$	$d^k(3)$	$d^k(4)$	$d^k(5)$	$d^k(6)$
0	0	0	0	0	0	0
1	-2 (2)	-5 (4)	-3 (5)	-5 (6)	-5 (4)	0
2	-7 (2)	-10 (4)	-8 (5)	-6 (3)	-10 (4)	-2 (5)
3	-12 (2)	-11 (4)	-13 (5)	-11 (3)	-11 (4)	-7 (5)
4	-15 (3)	-16 (4)	-14 (5)	-16 (3)	-16 (4)	-8 (5)
5	-18 (2)	-21 (4)	-19 (5)	-17 (3)	-21 (4)	-13 (5)
6	-23 (2)	-22 (4)	-24 (5)	-22 (3)	-22 (4)	-18 (5)

All the labels decrease in pass 6, so $j = 1$. Reading backwards: row 6 gives 2, row 5 for node 2 gives 4, then 3, 5, 4, 6. The walk 6, 4, 5, 3, 4, 2, 1 meets node 4 twice: $W = 4, 5, 3, 4$, with $c(W) = -11$ and $r(W) = 1$. The new flow is $x^1 = (1, 4, 4, 1, 3, 0, 3, 2, 1)$, of cost 69.

Iteration 2

The arcs of W change: (4, 5) and (5, 3) lose one unit, (3, 4) disappears, and the forward arcs (3, 5), (5, 4) appear. $G(x^1)$ is:

arc (i, j)	(1, 2)	(1, 3)	(2, 4)	(3, 2)	(3, 5)	(4, 3)	(4, 6)	(5, 4)	(5, 6)
x_{ij}^1	1	4	4	1	3	0	3	2	1
forward (i, j) : r, c	3, 2	–	–	1, 2	1, 3	3, 3	2, 5	1, 5	1, 3
backward (j, i) : r, c	1, –2	4, –2	4, –5	1, –2	3, –3	–	3, –5	2, –5	1, –3

k	$d^k(1)$	$d^k(2)$	$d^k(3)$	$d^k(4)$	$d^k(5)$	$d^k(6)$
0	0	0	0	0	0	0
1	–2 (2)	–5 (4)	–3 (5)	–5 (6)	–5 (4)	0
2	–7 (2)	–10 (4)	–8 (5)	–5	–10 (4)	–2 (5)
3	–12 (2)	–10	–13 (5)	–7 (6)	–10	–7 (5)
4	–15 (3)	–12 (4)	–13	–12 (6)	–12 (4)	–7
5	–15	–17 (4)	–15 (5)	–12	–17 (4)	–9 (5)
6	–19 (2)	–17	–20 (5)	–14 (6)	–17	–14 (5)

Again $j = 1$, and the walk is 6, 4, 5, 6, 4, 2, 1: $W = 6, 4, 5, 6$, made of the backward arcs of (4, 6) and (5, 4) and of the forward arc (5, 6), with $c(W) = -5 - 5 + 3 = -7$ and $r(W) = \min\{3, 2, 1\} = 1$. The new flow is $\mathbf{x}^2 = (1, 4, 4, 1, 3, 0, 2, 1, 2)$, of cost 62.

Iteration 3

In $G(\mathbf{x}^2)$ the arc (5, 6) is saturated: only its backward arc (6, 5) remains, with $r = 2$, and the arcs (6, 4), (4, 5) have residual capacities 2 and 1, the forward arcs (4, 6), (5, 4) residual capacities 3 and 2. Bellman–Ford gives:

k	$d^k(1)$	$d^k(2)$	$d^k(3)$	$d^k(4)$	$d^k(5)$	$d^k(6)$
0	0	0	0	0	0	0
1	–2 (2)	–5 (4)	–3 (5)	–5 (6)	–5 (4)	0
2	–7 (2)	–10 (4)	–8 (5)	–5	–10 (4)	0
3	–12 (2)	–10	–13 (5)	–5	–10	0
4	–15 (3)	–11 (3)	–13	–5	–10	0
5	–15	–13 (1)	–13	–5	–10	0
6	–15	–13	–15 (2)	–5	–10	0

Only node 3 decreases in pass 6: $j = 3$, and the walk is 6, 4, 5, 3, 1, 2, 3. Node 3 is met twice: $W = 3, 1, 2, 3$, made of the backward arc of (1, 3), the forward arc (1, 2) and the backward arc of (3, 2), with $c(W) = -2 + 2 - 2 = -2$ and $r(W) = \min\{4, 3, 1\} = 1$. The new flow is $\mathbf{x}^3 = (2, 3, 4, 0, 3, 0, 2, 1, 2) = \tilde{\mathbf{x}}$, of cost 60.

Iteration 4

Bellman–Ford on $G(\tilde{\mathbf{x}})$ repeats the rows 0–5 of iteration 3, and in pass 6 no label decreases: $d^6 = d^5 = (-15, -13, -13, -5, -10, 0)$. The algorithm stops with $\tilde{\mathbf{x}}$, certified by $\boldsymbol{\pi} = -d^6 = (15, 13, 13, 5, 10, 0)$, already normalized with $\pi_6 = 0$: the potentials of Chapter 4.

- In summary:

iteration	cost of $\mathbf{x}$	j	cycle W	$c(W)$	$r(W)$
1	80	1	4, 5, 3, 4	-11	1
2	69	1	6, 4, 5, 6	-7	1
3	62	3	3, 1, 2, 3	-2	1
4	60	–	no negative cycle	–	–

- The bound of [Theorem 6.14](#) allows $80 - 60 = 20$ iterations; three are enough. The three cycles are the cycles of the decomposition of [Example 6.2](#), found in a different order.

The cycle-canceling algorithm starts from a feasible flow and cancels negative cycles of $G(\mathbf{x})$, found by Bellman–Ford from labels all equal to 0. With integer data every cancellation lowers the cost by at least 1; at the stop the labels give the potentials that certify the optimal flow.

6.7 The successive shortest path algorithm

Definition 6.15: pseudoflow and imbalances

A **pseudoflow** is a vector $\mathbf{x}$ with $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$, which may violate flow conservation. The **imbalance** of node i is

$$e_i = b_i - \left(\sum_{(i,j) \in \delta^+(i)} x_{ij} - \sum_{(j,i) \in \delta^-(i)} x_{ji} \right) :$$

the supply that i has still to send if $e_i > 0$ (an **excess node**), the demand that it has still to receive if $e_i < 0$ (a **deficit node**).

- A pseudoflow is a feasible flow exactly when $\mathbf{e} = \mathbf{0}$. The imbalances add up to zero, since the balances do and every arc adds x_{ij} to its tail and subtracts it from its head.
- The residual network $G(\mathbf{x})$ of a pseudoflow is defined as in [Definition 6.1](#). Augmenting $\mathbf{x}$ along a path P of $G(\mathbf{x})$ from s to t by $\delta \leq r(P) = \min_{(i,j) \in P} r_{ij}$ is defined as for cycles: the bounds still hold, e_s decreases by δ , e_t increases by δ , and the cost grows by $\delta c(P)$, where $c(P) = \sum_{(i,j) \in P} c_{ij}^x$.
- The idea is to keep potentials with non-negative reduced costs on $G(\mathbf{x})$, the condition of [Theorem 6.9](#), while the imbalances go to zero. With $\mathbf{x} = \mathbf{0}$, $\boldsymbol{\pi} = \mathbf{0}$ and costs $c_{ij} \geq 0$ the condition holds at the start.

Lemma 6.16: shortest paths keep the reduced costs non-negative

Let $\mathbf{x}$ be a pseudoflow and $\boldsymbol{\pi}$ potentials with $\bar{c}_{ij}^\pi = c_{ij}^x - \pi_i + \pi_j \geq 0$ on every arc of $G(\mathbf{x})$. Let $d(i)$ be the distances from a node s in $G(\mathbf{x})$ with the lengths $\bar{c}^\pi$ ($d(i) = +\infty$ if i is not reachable), t a node with $d(t) < +\infty$, P a shortest path from s to t , and

$$\pi'_i = \pi_i - \min\{d(i), d(t)\}, \quad \forall i \in N.$$

- (a) $c_{ij}^x - \pi'_i + \pi'_j \geq 0$ on every arc of $G(\mathbf{x})$.

- (b) $c_{ij}^x - \pi'_i + \pi'_j = 0$ on every arc of P .
- (c) If $\mathbf{x}'$ is $\mathbf{x}$ augmented along P by $\delta \leq r(P)$, then $c_{ij}^{x'} - \pi'_i + \pi'_j \geq 0$ on every arc of $G(\mathbf{x}')$.

Proof. Let $\hat{d}(i) = \min\{d(i), d(t)\}$. The reduced cost of (i, j) with $\boldsymbol{\pi}'$ is $\bar{c}_{ij}^{\pi'} + \hat{d}(i) - \hat{d}(j)$.

- (a) Let (i, j) be an arc of $G(\mathbf{x})$. If $d(i) < +\infty$, a shortest path to i followed by (i, j) gives $d(j) \leq d(i) + \bar{c}_{ij}^{\pi}$, so $\hat{d}(j) \leq \min\{d(i) + \bar{c}_{ij}^{\pi}, d(t)\} \leq \hat{d}(i) + \bar{c}_{ij}^{\pi}$, since $\bar{c}_{ij}^{\pi} \geq 0$. If $d(i) = +\infty$, then $\hat{d}(i) = d(t) \geq \hat{d}(j)$. In both cases the new reduced cost is non-negative.
- (b) Subpaths of P are shortest (Lemma «optimality of subpaths» of Chapter 3), so $d(j) = d(i) + \bar{c}_{ij}^{\pi}$ on every arc of P , and $d(i) \leq d(t)$ along P since the lengths are non-negative. Hence $\hat{d} = d$ on P , and the new reduced cost is 0.
- (c) Augmenting along P changes $G(\mathbf{x})$ only on the arcs of P : an arc (i, j) of P may disappear, and the reverse arc (j, i) may appear. The reverse of a residual arc has the opposite cost, hence by (b) reduced cost 0; the other arcs keep their reduced costs, non-negative by (a). $\square$

- For every path P from s to t the potentials telescope: $\sum_{(i,j) \in P} \bar{c}_{ij}^{\pi} = c(P) - \pi_s + \pi_t$. A shortest path for the reduced costs is a minimum cost path of $G(\mathbf{x})$: the flow travels along cheapest paths, hence the name of the algorithm.

Algorithm

Algorithm: Successive shortest path algorithm

Input: a balanced network with $\mathbf{l} = \mathbf{0}$, integer b_i and u_{ij} , costs $c_{ij} \geq 0$

Output: an optimal flow $\mathbf{x}$ with its certificate $\boldsymbol{\pi}$, or «infeasible»

- 1 **Step 0 (initialization).** Set $\mathbf{x} := \mathbf{0}$, $\boldsymbol{\pi} := \mathbf{0}$ and $\mathbf{e} := \mathbf{b}$.
- 2 **Step 1 (selection).** If $\mathbf{e} = \mathbf{0}$, STOP: $\mathbf{x}$ is optimal, with $\boldsymbol{\pi}$.
- 3 Otherwise let s be the node of smallest index with $e_s > 0$.
- 4 **Step 2 (shortest paths).** Run Dijkstra from s on $G(\mathbf{x})$, lengths $c_{ij}^x - \pi_i + \pi_j$.
- 5 If no node with $e_t < 0$ is reachable, STOP: infeasible.
- 6 Otherwise t is the reachable node with $e_t < 0$ and smallest $d(t)$, and P its path.
- 7 **Step 3 (augmentation).** Set $\pi_i := \pi_i - \min\{d(i), d(t)\}$ for every $i \in N$.
- 8 Augment $\mathbf{x}$ along P by $\delta := \min\{e_s, -e_t, r(P)\}$, and update e_s, e_t .
- 9 Go to Step 1.

- **Rules of this chapter.** Dijkstra's algorithm breaks ties by the smallest index, as in Chapter 3; among the deficit nodes at the same distance, t is the one of smallest index.

Theorem 6.17: correctness of the successive shortest path algorithm

With the input of the algorithm, at the beginning of every iteration $\mathbf{x}$ is an integer pseudoflow and $c_{ij}^x - \pi_i + \pi_j \geq 0$ on every arc of $G(\mathbf{x})$. The algorithm stops after at most $D = \sum_{i: b_i > 0} b_i$ iterations. If it stops in Step 1, $\mathbf{x}$ is an optimal flow; if it stops in Step 2, the network has no feasible flow.

Proof. We prove the invariant, then the bound and the two stops.

1. *Invariant.* At the start $G(\mathbf{0})$ has the forward arcs only, with costs $c_{ij} \geq 0$, and $\boldsymbol{\pi} = \mathbf{0}$. Dijkstra's algorithm applies, the lengths being non-negative, and Lemma 6.16(c) keeps the reduced costs non-negative. The amount δ is a positive integer, since e_s , $-e_t$ and the residual capacities are positive integers (or $+\infty$), so $\mathbf{x}$ stays integer.
2. *Bound.* If $e_s > 0$, some node has $e_t < 0$, since the imbalances add up to zero. Every iteration lowers $\sum_{i: e_i > 0} e_i$ by $\delta \geq 1$, and no imbalance changes sign, since $\delta \leq e_s$ and $\delta \leq -e_t$. The sum starts at D .
3. *Stop in Step 1.* $\mathbf{e} = \mathbf{0}$, so $\mathbf{x}$ is a feasible flow, optimal by the invariant and Theorem 6.9.
4. *Stop in Step 2.* Let S be the set of the nodes reachable from s . No arc of $G(\mathbf{x})$ leaves S , so $x_{ij} = u_{ij}$ on $\delta^+(S)$ and $x_{ij} = 0$ on $\delta^-(S)$. The pseudoflow satisfies conservation with the balances $b_i - e_i$, so by Lemma «flow across a cut» of Chapter 4

$$\sum_{i \in S} b_i = x(\delta^+(S)) - x(\delta^-(S)) + \sum_{i \in S} e_i = u(\delta^+(S)) + \sum_{i \in S} e_i > u(\delta^+(S)),$$

because $e_s > 0$ and no node of S has $e_i < 0$. The cut condition of Chapter 4 fails for S : there is no feasible flow. $\square$

- **Running time.** Each iteration runs Dijkstra's algorithm in $O(n^2)$ time: $O(Dn^2)$ in total, a pseudo-polynomial bound.
- **Negative costs.** If some arcs with finite capacity have $c_{ij} < 0$, start from $x_{ij} = u_{ij}$ on them and $x_{ij} = 0$ elsewhere, with $\mathbf{e} = \mathbf{b} - \mathbf{B}\mathbf{x}$: such an arc gives only its backward arc, of cost $-c_{ij} > 0$, and $\boldsymbol{\pi} = \mathbf{0}$ works again.

6.7.1 Successive shortest paths on the reference network

Example 6.7: successive shortest paths

We start from $\mathbf{x} = \mathbf{0}$, $\boldsymbol{\pi} = \mathbf{0}$ and $\mathbf{e} = \mathbf{b} = (5, 2, 0, -3, 0, -4)$. In the first table each row is an iteration, with the node s , the distances of Dijkstra's algorithm and the node t ; the second table gives the path P , its reduced cost $d(t)$ and its cost $c(P)$, the amount δ and the cost of the new flow.

it.	s	$d(1)$	$d(2)$	$d(3)$	$d(4)$	$d(5)$	$d(6)$	t
1	1	0	2	2	7	5	8	4
2	1	0	0	0	0	0	1	6
3	2	0	0	0	0	0	4	6
4	2	0	0	0	3	0	3	6

it.	P	$d(t)$	$c(P)$	δ	cost
1	1, 2, 4	7	7	3	21
2	1, 3, 5, 6	1	8	2	37
3	2, 4, 6	4	10	1	47
4	2, 1, 3, 5, 4, 6	3	13	1	60

The third table gives, after each iteration, the potentials, the imbalances and the flow.

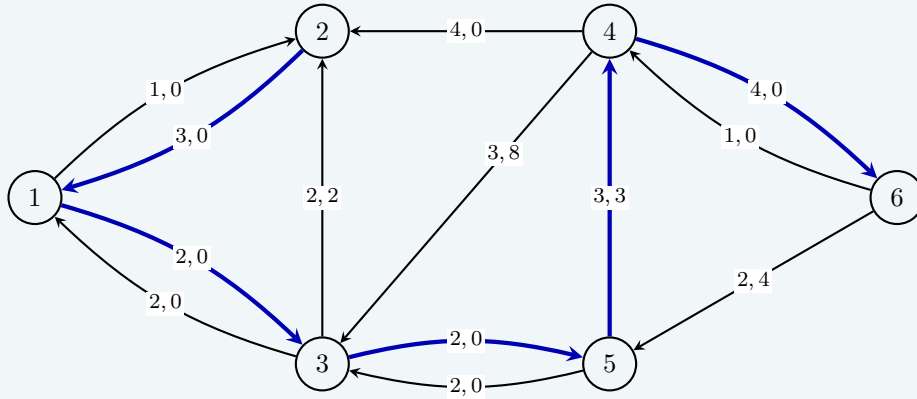
it.	$\boldsymbol{\pi}$	$\mathbf{e}$	$\mathbf{x}$
0	(0, 0, 0, 0, 0, 0)	(5, 2, 0, -3, 0, -4)	(0, 0, 0, 0, 0, 0, 0, 0)
1	(0, -2, -2, -7, -5, -7)	(2, 2, 0, 0, 0, -4)	(3, 0, 3, 0, 0, 0, 0, 0)
2	(0, -2, -2, -7, -5, -8)	(0, 2, 0, 0, 0, -2)	(3, 2, 3, 0, 2, 0, 0, 0, 2)
3	(0, -2, -2, -7, -5, -12)	(0, 1, 0, 0, 0, -1)	(3, 2, 4, 0, 2, 0, 1, 0, 2)
4	(0, -2, -2, -10, -5, -15)	(0, 0, 0, 0, 0, 0)	(2, 3, 4, 0, 3, 0, 2, 1, 2)

The reduced costs $\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j$ of the arcs of A , before each iteration, are (the forward arc has $\bar{c}_{ij}$ if $x_{ij} < u_{ij}$, the backward arc $-\bar{c}_{ij}$ if $x_{ij} > 0$):

it.	(1, 2)	(1, 3)	(2, 4)	(3, 2)	(3, 5)	(4, 3)	(4, 6)	(5, 4)	(5, 6)
1	2	2	5	2	3	3	5	5	3
2	0	0	0	2	0	8	5	3	1
3	0	0	0	2	0	8	4	3	0
4	0	0	0	2	0	8	0	3	-4

- *Iteration 1.* The reduced costs are the costs. The deficit nodes 4 and 6 are at distance 7 and 8: $t = 4$, and $\delta = \min\{5, 3, 4, 4\} = 3$. Node 6 is farther than t and its potential decreases by 7, not 8.
- *Iteration 2.* The arcs (1, 2), (2, 4) of the first path now have reduced cost 0, and so do their backward arcs. The path 1, 3, 5, 6 has reduced cost 1 and cost 8; $\delta = \min\{2, 4, 4, 4, 2\} = 2$ is set by e_1 and by u_{56} .

- *Iteration 3.* $s = 2$, and $d(1) = 0$ through the backward arc $(2, 1)$. The path $2, 4, 6$ uses the last unit of $(2, 4)$.
- *Iteration 4.* The path $2, 1, 3, 5, 4, 6$ uses the backward arc $(2, 1)$: one unit that went from 1 to 2 is rerouted from 1 through 3, 5 and 4. The figure shows $G(\mathbf{x})$ with r_{ij} and the reduced cost on each arc, and the path in blue.



After iteration 4 the imbalances are zero and $\mathbf{x} = \tilde{\mathbf{x}}$, of cost 60. Shifted by 15, the potentials become $(15, 13, 13, 5, 10, 0)$: the potentials of Chapter 4.

- Here the costs 7, 8, 10, 13 of the paths grow from one iteration to the next: later units travel along more expensive paths, and the total cost is $3 \cdot 7 + 2 \cdot 8 + 10 + 13 = 60$.
- Four iterations are enough here, while the bound of [Theorem 6.17](#) is $D = 7$.

Dijkstra needs the reduced costs

In iteration 4 the arcs $(2, 1)$, $(3, 1)$, $(4, 2)$, $(5, 3)$, $(6, 4)$, $(6, 5)$ of $G(\mathbf{x})$ are backward arcs with negative costs c_{ij}^x , and Dijkstra's algorithm with negative lengths can fail ([Chapter 3](#)). The potentials make all the lengths non-negative without changing the shortest paths.

The successive shortest path algorithm sends flow from an excess node to a deficit node along a shortest path of $G(\mathbf{x})$, found by Dijkstra's algorithm on the reduced costs. The update $\pi_i := \pi_i - \min\{d(i), d(t)\}$ keeps the reduced costs non-negative, so the final flow is optimal and the potentials certify it.

6.8 Typical mistakes

Warning

- **Giving the backward arc the cost $+c_{ij}$.** A backward arc decreases x_{ij} , so it saves c_{ij} per unit: its cost is $-c_{ij}$, and its residual capacity is $x_{ij} - l_{ij}$.
- **Augmenting by more than $r(W)$.** Some arc of the cycle would leave its bounds; $\theta = r(W)$ is the largest step.
- **Running Bellman–Ford from a single node.** A negative cycle that is not reachable from that node is missed; the labels start at 0 on every node.
- **Taking the walk for the cycle.** The walk with n arcs read from the table contains a negative cycle, but it is not a cycle itself: the cycle is the piece between two occurrences of the same node.
- **Running Dijkstra’s algorithm on the costs c_{ij}^x .** Backward arcs can have negative costs; the lengths are the reduced costs $c_{ij}^x - \pi_i + \pi_j$.
- **Updating the potentials with the wrong sign.** With $\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j$ the update is $\pi_i := \pi_i - \min\{d(i), d(t)\}$; the nodes farther than t decrease by $d(t)$ only.
- **Taking the maximum flow of Step 0 as optimal.** The auxiliary network has no costs: its maximum flow is feasible, and optimal only by chance.

6.9 In summary

Summary

- The residual network $G(\mathbf{x})$ has a forward arc (i, j) with $r_{ij} = u_{ij} - x_{ij}$ and cost c_{ij} , and a backward arc (j, i) with $r_{ji} = x_{ij} - l_{ij}$ and cost $-c_{ij}$; augmenting along a cycle W changes the cost by $\theta c(W)$.
- A circulation decomposes into cycles; the difference of two feasible flows is a circulation on $G(\mathbf{x})$.
- A feasible flow is optimal if and only if $G(\mathbf{x})$ has no negative cycle, if and only if some potentials give non-negative reduced costs on $G(\mathbf{x})$; on the arcs of A : $\bar{c}_{ij} > 0 \Rightarrow x_{ij} = l_{ij}$, $\bar{c}_{ij} < 0 \Rightarrow x_{ij} = u_{ij}$.
- A maximum flow from a super source to a super sink finds a feasible flow or proves that there is none.
- Cycle canceling: Bellman–Ford from labels 0 finds a negative cycle, augmenting by $r(W)$ cancels it; at most $\mathbf{c}'\mathbf{x}^0 - z(\text{MCF})$ iterations with integer data.
- Successive shortest paths: Dijkstra on the reduced costs from an excess node to the nearest deficit node, $\pi_i := \pi_i - \min\{d(i), d(t)\}$; at most D iterations.

The reference network at a glance.

most expensive flow	$\mathbf{x}^0 = (1, 4, 4, 1, 4, 1, 3, 3, 1)$, cost 80
cycle canceling from $\mathbf{x}^0$	cycles 4, 5, 3, 4; 6, 4, 5, 6; 3, 1, 2, 3: costs 80, 69, 62, 60
successive shortest paths	paths 1, 2, 4; 1, 3, 5, 6; 2, 4, 6; 2, 1, 3, 5, 4, 6: costs 21, 37, 47, 60
first feasible flow	maximum flow of value 7; on A it is $\tilde{\mathbf{x}}$
optimal flow	$\tilde{\mathbf{x}} = (2, 3, 4, 0, 3, 0, 2, 1, 2)$, $z(\text{MCF}) = 60$
potentials ($\pi_6 = 0$)	$\boldsymbol{\pi} = (15, 13, 13, 5, 10, 0)$

Transportation and assignment

Prerequisites

- From this module, [Chapter 1 «Graphs»](#): trees, leaves, fundamental cycles and the rank of the node–arc incidence matrix.
- From this module, [Chapter 3 «Shortest paths»](#): distances, and [Chapter 4 «Network flow models»](#): the minimum cost flow model, the dummy node, integer vertices, potentials and reduced costs.
- From the module Linear Programming, [Chapter 9 «Duality theory»](#): the dual of an LP model and the weak and strong duality theorems.

Objectives

At the end of this chapter you will be able to:

- write the transportation model and its tableau, balance an instance with a dummy origin or destination, and reduce an uncapacitated transshipment instance to a transportation instance;
- recognize the bases of the transportation model as the spanning trees of a bipartite network, compute a basic solution and recognize a degenerate one;
- build an initial basic feasible solution with the north-west corner rule and the minimum cost rule;
- run the potential method, with potentials, reduced costs, cycles and pivots, and certify the optimality of a solution with the dual model;
- write the assignment model, prove the theorem of König and Egerváry, and run the Hungarian algorithm, reading its potentials as an optimal dual solution.

7.1 The transportation problem

Definition 7.1: transportation problem

There are m **origins** and n **destinations**. We number the origins $1, 2, \dots, m$ and the destinations $m + 1, m + 2, \dots, m + n$, and we write

$$P = \{1, 2, \dots, m\}, \quad D = \{m + 1, m + 2, \dots, m + n\}.$$

- Every origin $i \in P$ has a **supply** $a_i > 0$, every destination $j \in D$ has a **demand** $d_j > 0$, and $c_{ij} \in \mathbb{R}$ is the cost of shipping one unit from i to j .
- The instance is **balanced** if $\sum_{i \in P} a_i = \sum_{j \in D} d_j$.

The **transportation problem** asks for the amounts $x_{ij} \geq 0$ to ship from every origin i to every destination j , so that every origin ships its whole supply and every destination receives its demand, at minimum total cost.

- In this chapter m and n are the numbers of origins and destinations. The underlying network has $m + n$ nodes and one arc (i, j) for every $i \in P$ and $j \in D$, that is $m n$ arcs: it is the **complete bipartite network** of P and D , since every arc goes from P to D .
- Numbering the destinations after the origins gives every node its own number, and later its own potential.
- We use one continuous variable x_{ij} for every pair $i \in P, j \in D$: the amount shipped from i to j .

Using these variables, an LP model for the transportation problem reads as follows:

$$\min \sum_{i \in P} \sum_{j \in D} c_{ij} x_{ij} \quad (7.1a)$$

$$\text{subject to } \sum_{j \in D} x_{ij} = a_i, \quad \forall i \in P, \quad (7.1b)$$

$$\sum_{i \in P} x_{ij} = d_j, \quad \forall j \in D, \quad (7.1c)$$

$$x_{ij} \geq 0, \quad \forall i \in P, \forall j \in D. \quad (7.1d)$$

This LP model is referred to as the TP model.

- Description of the objective function and constraints of the TP model:
 - the objective function (7.1a) minimizes the total shipping cost;
 - constraints (7.1b), one for every origin, ship the whole supply of the origin;
 - constraints (7.1c), one for every destination, deliver the whole demand of the destination;
 - finally, constraints (7.1d) define the variables of the LP model.

- Multiplying the constraints (7.1c) by -1 gives the flow conservation constraints of the complete bipartite network, with balances $b_i = a_i$ for $i \in P$, $b_j = -d_j$ for $j \in D$, and no capacities. The TP model is the MCF model of the row «Transportation» of the table «Special cases» of [Chapter 4 «Network flow models»](#).

Proposition 7.2: feasibility and integrality

An instance of the transportation problem has a feasible solution if and only if it is balanced. A balanced instance has an optimal solution, and if the supplies a_i and the demands d_j are integer, it has an integer optimal solution.

Proof. Let $S = \sum_{i \in P} a_i$.

1. *Balance is necessary.* If $\mathbf{x}$ is feasible, adding up the constraints (7.1b) and the constraints (7.1c) gives the same sum of all the variables: $\sum_{i \in P} a_i = \sum_{i \in P} \sum_{j \in D} x_{ij} = \sum_{j \in D} d_j$.
2. *Balance is sufficient.* If $S = \sum_{j \in D} d_j$, the amounts $x_{ij} = a_i d_j / S \geq 0$ are feasible: row i adds up to $a_i \sum_{j \in D} d_j / S = a_i$, and column j to $d_j \sum_{i \in P} a_i / S = d_j$.
3. *Optimal solution.* Every feasible solution has $0 \leq x_{ij} \leq a_i$, so the feasible region is a nonempty bounded polyhedron, and a linear function attains its minimum over it at a vertex ([Chapter 15 «The geometry of Linear Programming»](#) of the module Linear Programming).
4. *Integrality.* With integer balances and no capacities, every vertex is integer by the Theorem «integer vertices» of [Chapter 4](#). $\square$

7.1.1 The transportation tableau

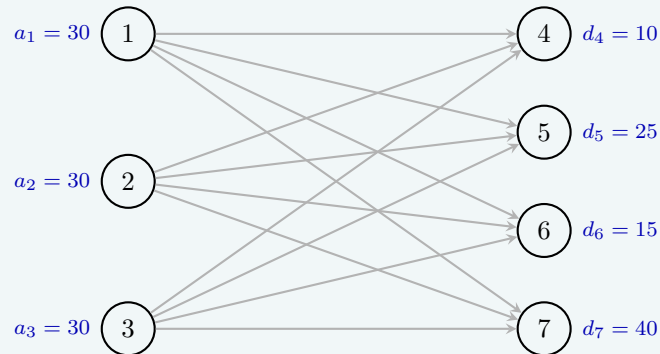
- The **transportation tableau** is an $m \times n$ table with one row for every origin and one column for every destination. The cell (i, j) holds the cost c_{ij} , or later the amount x_{ij} ; the last column holds the supplies a_i and the last row the demands d_j .
- A vector $\mathbf{x} \geq \mathbf{0}$ written in the tableau is feasible if and only if every row adds up to its supply and every column to its demand.

Example 7.1: three warehouses and four stores

A company has three warehouses, the origins 1, 2, 3, with 30 units each, and four stores, the destinations 4, 5, 6, 7, with demands 10, 25, 15 and 40. The unit costs are in the tableau:

c_{ij}	4	5	6	7	a_i
1	1	9	2	3	30
2	3	4	2	2	30
3	9	1	4	6	30
d_j	10	25	15	40	90

The instance is balanced: the total supply and the total demand are both 90. In the complete bipartite network every origin is joined to every destination, with $3 \cdot 4 = 12$ arcs; next to each node, its supply or its demand.



The proportional amounts $x_{ij} = a_i d_j / 90$ of Proposition 7.2 are feasible, for example $x_{14} = 30 \cdot 10 / 90 = \frac{10}{3}$, but fractional; the vertices of the feasible region are integer.

7.1.2 Unbalanced instances

- If the total supply exceeds the total demand and every origin ships *at most* its supply, we add a **dummy destination** with demand $\sum_{i \in P} a_i - \sum_{j \in D} d_j$ and cost 0 from every origin. The amount shipped to it is the unused supply of the origin: this is the dummy node of Chapter 4, as a column of the tableau.
- If the total demand exceeds the total supply and a destination may receive less than its demand, we add a **dummy origin** with supply $\sum_{j \in D} d_j - \sum_{i \in P} a_i$. The cost of its cell $(\cdot, j)$ is the penalty for each unit of unmet demand of j , or 0 if there is none.

Example 7.2: a dummy destination

In Example 7.1 the supply of warehouse 1 grows to $a_1 = 40$: the total supply is 100 and the total demand 90.

- We add the dummy destination 8 with demand $d_8 = 100 - 90 = 10$, and the cells $(1, 8)$, $(2, 8)$, $(3, 8)$ with cost 0: the tableau has a fifth column.
- The optimal solution, computed with Gurobi, is unique:

$$\tilde{x}_{14} = 10, \tilde{x}_{16} = 15, \tilde{x}_{17} = 10, \tilde{x}_{18} = 5, \tilde{x}_{27} = 30, \tilde{x}_{35} = 25, \tilde{x}_{38} = 5,$$

with cost 155. Warehouses 1 and 3 keep 5 units each, and warehouse 2 ships all its supply.

7.1.3 Transshipment as transportation

- In a transshipment instance the units may pass through intermediate nodes. When the arcs have no capacities, every unit can travel along a shortest path, and only the distances between supply and demand nodes matter.

Proposition 7.3: transshipment as transportation

Let $G = (N, A)$ be a network with balances b_i , zero lower bounds, no capacities and costs $c_{ij} \geq 0$. Let $P = \{i \in N : b_i > 0\}$ and $D = \{j \in N : b_j < 0\}$, and assume that every $j \in D$ is reachable from every $i \in P$, with distance $\tilde{d}(i, j)$ for the lengths c_{ij} . Then the minimum cost of a feasible flow equals the optimal value of the transportation instance with origins P , supplies $a_i = b_i$, destinations D , demands $d_j = -b_j$ and costs $\tilde{d}(i, j)$.

Proof. We compare the two optimal values.

1. *From shipments to flows.* Let $\mathbf{y}$ be feasible for the transportation instance. Sending y_{ij} units along a shortest path from i to j , for every pair, gives a flow $\mathbf{x} \geq \mathbf{0}$: node $i \in P$ sends $\sum_{j \in D} y_{ij} = b_i$ units, node $j \in D$ receives $-b_j$, the other nodes pass on what they receive. Its cost is $\sum_{i,j} \tilde{d}(i, j) y_{ij}$, so the minimum flow cost is at most the transportation optimum.
2. *From flows to shipments.* We show by induction on the number of arcs with $x_{ij} > 0$ plus the number of nodes with $b_i \neq 0$ that every feasible flow $\mathbf{x}$ costs at least $\sum_{i,j} \tilde{d}(i, j) y_{ij}$ for some feasible $\mathbf{y}$. If $\mathbf{b} = \mathbf{0}$, take $\mathbf{y} = \mathbf{0}$.
3. Otherwise start at a node i_0 with $b_{i_0} > 0$: its outflow is positive, so it has a leaving arc with positive flow. A node v with $b_v \geq 0$ reached along such an arc has a positive inflow, hence a positive outflow. Follow these arcs until a node repeats or a node j_0 with $b_{j_0} < 0$ is reached.
4. If a node repeats, the arcs form a directed cycle C with positive flows. Subtracting their minimum ε along C keeps the flow feasible, does not increase the cost, since $c \geq 0$, and empties an arc: the induction applies.
5. Otherwise we have a path Q from i_0 to j_0 . Let $\varepsilon > 0$ be the minimum of b_{i_0} , $-b_{j_0}$ and the flows on Q . Subtracting ε along Q , and from b_{i_0} and $-b_{j_0}$, gives a feasible flow $\mathbf{x}'$ for the new balances, with fewer positive arcs or nonzero balances, and

$$\mathbf{c}'\mathbf{x} = \mathbf{c}'\mathbf{x}' + \varepsilon c(Q) \geq \mathbf{c}'\mathbf{x}' + \varepsilon \tilde{d}(i_0, j_0).$$

By induction $\mathbf{c}'\mathbf{x}' \geq \sum \tilde{d}(i, j) y'_{ij}$; adding ε to $y'_{i_0 j_0}$ gives the required $\mathbf{y}$. $\square$

Example 7.3: the transshipment instance of Chapter 4 without capacities

We take the reference network of Chapter 4, with the same balances $\mathbf{b} = (5, 2, 0, -3, 0, -4)$ and costs, and remove the capacities. The origins are the supply nodes 1, 2 and the destinations the demand nodes 4, 6; we keep the names of the network.

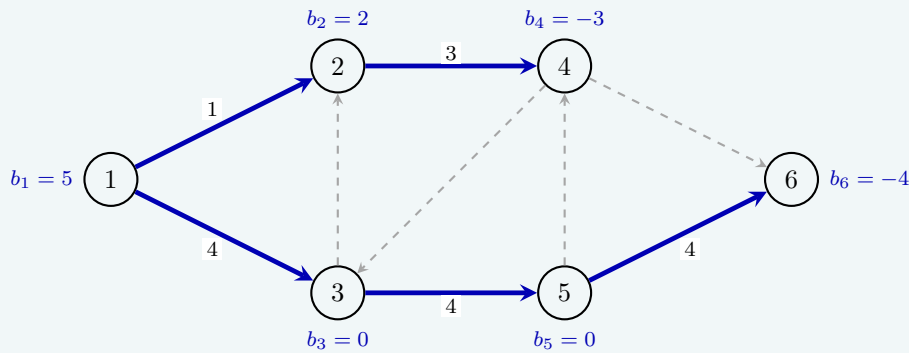
- The distances are $\tilde{d}(1, 4) = 7$ along 1, 2, 4, $\tilde{d}(1, 6) = 8$ along 1, 3, 5, 6, $\tilde{d}(2, 4) = 5$ along 2, 4, and $\tilde{d}(2, 6) = 10$ along 2, 4, 6.

- The 2×2 transportation instance has $a_1 = 5$, $a_2 = 2$, $d_4 = 3$, $d_6 = 4$. With $y_{24} = t \in [0, 2]$ the other amounts are $y_{14} = 3 - t$, $y_{16} = 2 + t$, $y_{26} = 2 - t$, and the cost is

$$7(3 - t) + 8(2 + t) + 5t + 10(2 - t) = 57 - 4t,$$

minimum for $t = 2$: $y_{14} = 1$, $y_{16} = 4$, $y_{24} = 2$, $y_{26} = 0$, with cost 49.

- Along the shortest paths this gives the flow $x_{12} = 1$, $x_{13} = 4$, $x_{24} = 3$, $x_{35} = 4$, $x_{56} = 4$ (blue), of cost 49, the optimum computed by Gurobi. With the capacities of Chapter 4 this flow is not feasible, since the arc (5, 6) has capacity 2, and the optimum is 60.



The transportation problem ships the supplies of m origins to the demands of n destinations at minimum cost. It is a minimum cost flow model on the complete bipartite network; it is feasible exactly when it is balanced, and a dummy origin or destination balances it.

7.2 Basic solutions

- The column of x_{ij} in the constraints of the TP model has two entries equal to 1, in the row of the supply of i and in the row of the demand of j . Changing the sign of the demand rows gives the incidence matrix $\mathbf{B}$ of the complete bipartite network, and changes neither the rank nor the linear dependences of the columns.

Proposition 7.4: rank of the transportation model

The $m + n$ constraints of the TP model have rank $m + n - 1$. In a balanced instance each constraint is implied by the others.

Proof. Every origin is joined to every destination, so the complete bipartite network is connected, and its incidence matrix has rank $m + n - 1$ by the Theorem «rank of the incidence matrix» of Chapter 1. The sum of the supply rows minus the sum of the demand rows is zero, since both sums contain every variable once, and the same combination of the right-hand sides is $\sum_{i \in P} a_i - \sum_{j \in D} d_j = 0$: any constraint is a

combination of the others. □

Definition 7.5: basis and basic solution

A **basis** of the TP model is a set T of $m + n - 1$ cells whose arcs form a spanning tree of the complete bipartite network; its cells are the **basic cells**. A **basic solution** of T is a solution of the constraints (7.1b) and (7.1c) with $x_{ij} = 0$ for every cell outside T . It is a **basic feasible solution** if $\mathbf{x} \geq \mathbf{0}$, and it is **degenerate** if some basic cell has $x_{ij} = 0$.

Theorem 7.6: bases are spanning trees

- (a) A set of $m + n - 1$ cells has linearly independent columns if and only if its arcs form a spanning tree.
- (b) In a balanced instance, every basis T has exactly one basic solution. It is found by fixing, again and again, the cell of a leaf of the tree, and it is integer when the supplies and the demands are integer.

Proof. Part (a) is the Corollary «bases of the incidence matrix» of Chapter 1, since the sign of the demand rows does not change the dependences of the columns. For part (b):

1. *Uniqueness.* Two basic solutions of T differ by a vector $\mathbf{y}$ that is zero outside T and satisfies the constraints with zero right-hand sides; the columns of T are independent by (a), so $\mathbf{y} = \mathbf{0}$.
2. *Construction.* The tree has a leaf v (Lemma «leaves» of Chapter 1), with only one cell e of T . The constraint of v contains no other basic cell, so x_e equals the supply or demand of v .
3. We remove v and e , and subtract x_e from the supply or demand of the other endpoint of e . The remaining cells form a spanning tree of the remaining nodes, and the remaining supplies and demands still have equal sums.
4. When a single cell is left, both its endpoints are leaves, and the equal sums give the same value for both constraints. Every value is obtained by sums and differences of the data, so it is integer for integer data. □

Proposition 7.7: vertices are basic

Every vertex $\bar{\mathbf{x}}$ of the feasible region of the TP model is a basic feasible solution: its cells with $\bar{x}_{ij} > 0$ form a forest, contained in a basis T , and $\bar{\mathbf{x}}$ is the basic solution of T .

Proof. Let F be the cells with $\bar{x}_{ij} > 0$.

1. The columns of F are linearly independent, by the first step of the proof of the Theorem «integer vertices» of Chapter 4: here the cells strictly between their bounds are those of F . A cycle has dependent columns (Lemma «columns of a

cycle» of Chapter 1), so F is a forest.

2. The network is connected: while the forest has two or more components, some arc joins one of them to another one, and adding it creates no cycle (Lemma «adding an edge» of Chapter 1). We end with a spanning tree $T \supseteq F$.
3. $\bar{x}$ is zero outside $F \subseteq T$, so it is the basic solution of T , unique by Theorem 7.6. $\square$

- A cycle of the bipartite network alternates origins and destinations. In the tableau it is a closed path through cells of the set, moving alternately along a row and along a column: a set of cells is a basis if it has $m + n - 1$ cells and no such closed path.
- A basic feasible solution has at most $m + n - 1$ positive amounts; it has fewer when it is degenerate.

Example 7.4: bases of the transportation instance

In Example 7.1 a basis has $m + n - 1 = 3 + 4 - 1 = 6$ cells.

- The cells $(1, 4), (1, 5), (2, 4), (2, 5), (3, 6), (3, 7)$ are not a basis: $(1, 4), (2, 4), (2, 5), (1, 5)$ form the cycle $1, 4, 2, 5, 1$, and indeed $A_{14} - A_{24} + A_{25} - A_{15} = \mathbf{0}$ for their columns A_{ij} .
- The cells $(1, 4), (1, 5), (1, 6), (1, 7), (2, 7), (3, 7)$ form a spanning tree. Its leaves 2, 3, 4, 5, 6 fix $x_{27} = 30, x_{37} = 30, x_{14} = 10, x_{15} = 25, x_{16} = 15$; then the row of origin 1 gives

$$x_{17} = 30 - 10 - 25 - 15 = -20.$$

The basic solution exists, but it is not feasible.

The constraints of the TP model have rank $m + n - 1$. A basis is a spanning tree of the bipartite network, with $m + n - 1$ cells; its basic solution is found leaf by leaf, it is integer for integer data, and every vertex is a basic feasible solution.

7.3 Initial basic feasible solutions

- A basic feasible solution is built by filling the tableau one cell at a time. A row or a column is **active** until it is crossed out; the residual supply $\bar{a}_i$ and the residual demand $\bar{d}_j$ are what is left to ship and to receive.
- The two rules below differ only in the choice of the next cell.

Algorithm

Algorithm: Initial basic feasible solution

Input: a balanced instance with supplies a_i , demands d_j and costs c_{ij}

Output: a basis T and its basic feasible solution $\bar{x}$

- 1 **Step 0 (initialization).** All rows and columns are active; $\bar{a}_i := a_i$, $\bar{d}_j := d_j$, $T := \emptyset$.
- 2 **Step 1 (choice of the cell).** Choose a cell (i, j) with active row and column:
 - 3 *north-west corner rule:* the cell with the smallest i and the smallest j ;
 - 4 *minimum cost rule:* the cell with the smallest c_{ij} (the smallest i , then j , on ties).
- 5 **Step 2 (amount).** Set $\bar{x}_{ij} := \min\{\bar{a}_i, \bar{d}_j\}$, add (i, j) to T , and subtract $\bar{x}_{ij}$ from $\bar{a}_i$ and from $\bar{d}_j$.
- 6 **Step 3 (crossing out).** If $\bar{a}_i = 0$ and another row is active, cross out row i ; otherwise cross out column j .
- 7 If no column is active, **STOP**; otherwise go to Step 1.

- When $\bar{a}_i = \bar{d}_j = 0$ at the same time, only the row is crossed out: the column stays active with residual demand 0, and a later cell of that column gets the amount 0. This basic cell with value 0 keeps T a spanning tree, and the solution is degenerate.

Theorem 7.8: the initial solution is basic

With either rule, the algorithm «Initial basic feasible solution» stops after $m + n - 1$ steps, and T is a basis with basic feasible solution $\bar{x}$.

Proof. 1. *Invariant.* The residual supplies of the active rows and the residual demands of the active columns have equal sums: true at the start, and Step 2 subtracts the same amount from both sides.

2. *A crossed line has residual 0.* A crossed row has $\bar{a}_i = 0$. A column is crossed when $\bar{a}_i > 0$, and then $\bar{x}_{ij} = \bar{d}_j$ and $\bar{d}_j = 0$, or when row i is the only active row with $\bar{a}_i = 0$, and then every active column has residual 0 by the invariant.

3. *Counting.* A row is crossed out only if another row is active, so one row r is never crossed out, and the algorithm stops when the n columns are crossed out. At the stop the active rows have residual 0 by the invariant, and a row that reaches 0 without being crossed out is the only active row. Hence the steps are $(m - 1) + n = m + n - 1$, one cell each, all different, since a cell is chosen in two active lines and one of them is then crossed out.

4. *Feasibility.* At the stop every row and every column has residual 0, so $\bar{x} \geq 0$

satisfies all the constraints.

5. *Tree.* Let L_t be the line crossed at step t : no later cell lies in L_t . Backwards from the last step, the cells of steps $t, t+1, \dots$ form a tree: the cell of step t joins the new node L_t to its other line, which is still active, hence a node of the later cells or the row r . At $t = 1$ the tree spans all the $m + n$ nodes. $\square$

7.3.1 The north-west corner rule

Example 7.5: the north-west corner rule

On [Example 7.1](#) the north-west corner rule starts at the cell $(1, 4)$ and moves right or down. Its steps are:

step	cell	amount	crossed out
1	(1, 4)	10	column 4
2	(1, 5)	20	row 1
3	(2, 5)	5	column 5
4	(2, 6)	15	column 6
5	(2, 7)	10	row 2
6	(3, 7)	30	row 3, column 7

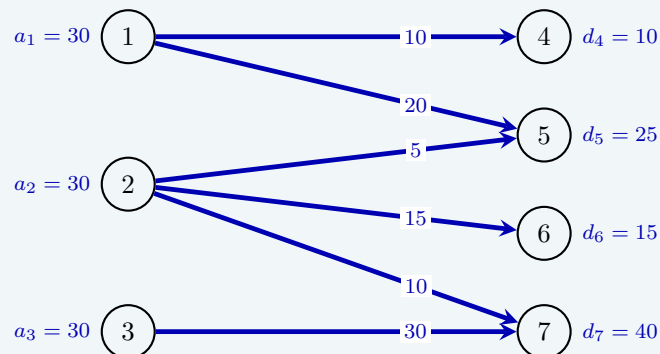
The basic feasible solution, with the basic cells in blue, is

$\bar{x}_{ij}$	4	5	6	7	a_i
1	10	20			30
2		5	15	10	30
3				30	30
d_j	10	25	15	40	

Its cost is

$$1 \cdot 10 + 9 \cdot 20 + 4 \cdot 5 + 2 \cdot 15 + 2 \cdot 10 + 6 \cdot 30 = 440.$$

The six basic cells form the spanning tree below, with the amounts on the arcs.



- The north-west corner rule ignores the costs: on this instance it ships 20 units on the cell (1, 5), of cost 9, and 30 units on (3, 7), of cost 6.

7.3.2 The minimum cost rule

Example 7.6: the minimum cost rule

On [Example 7.1](#) the minimum cost rule chooses, at every step, the cheapest cell of the active rows and columns:

step	cell	amount	crossed out
1	(1, 4)	10	column 4
2	(3, 5)	25	column 5
3	(1, 6)	15	column 6
4	(2, 7)	30	row 2
5	(1, 7)	5	row 1
6	(3, 7)	5	row 3, column 7

- At step 1 the cells (1, 4) and (3, 5) both cost 1, and (1, 4) goes first, since its row is smaller. At step 3 the cells (1, 6), (2, 6), (2, 7) cost 2, and (1, 6) goes first.
- At step 4 only column 7 is active: it takes the whole supply of row 2, then 5 units from row 1 and 5 from row 3.

The basic feasible solution is

$\bar{x}_{ij}$	4	5	6	7	a_i
1	10		15	5	30
2				30	30
3		25		5	30
d_j	10	25	15	40	

with cost

$$1 \cdot 10 + 2 \cdot 15 + 3 \cdot 5 + 2 \cdot 30 + 1 \cdot 25 + 6 \cdot 5 = 170.$$

Filling one cell per step and crossing out one line per step gives a basic feasible solution with $m + n - 1$ basic cells. The north-west corner rule ignores the costs; the minimum cost rule usually starts much closer to the optimum (170 instead of 440 on the example).

7.4 The potential method

- We write the constraints (7.1c) as $-\sum_{i \in P} x_{ij} = -d_j$, the conservation constraints of the destinations, and use a free dual variable π_v , the potential of node v , for the constraint of every node $v \in P \cup D$, as in Chapter 4.

Using a free dual variable π_v for the constraint of each node $v \in P \cup D$, the dual model of the TP model is:

$$\max \sum_{i \in P} a_i \pi_i - \sum_{j \in D} d_j \pi_j \quad (7.2a)$$

$$\text{subject to } \pi_i - \pi_j \leq c_{ij}, \quad \forall i \in P, \forall j \in D, \quad (7.2b)$$

$$\pi_v \in \mathbb{R}, \quad \forall v \in P \cup D. \quad (7.2c)$$

This LP model is referred to as the $D(\text{TP})$ model.

- With the reduced cost $\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j$ of Chapter 4, the dual constraint (7.2b) reads $\bar{c}_{ij} \geq 0$.
- Adding the same constant to all the potentials changes neither the reduced costs nor, in a balanced instance, the objective (7.2a). We normalize with $\pi_{m+n} = 0$, the potential of the last destination.
- For every $\mathbf{x}$ that satisfies the constraints (7.1b) and (7.1c), the potentials add up row by row and column by column:

$$\sum_{i \in P} \sum_{j \in D} (\pi_i - \pi_j) x_{ij} = \sum_{i \in P} a_i \pi_i - \sum_{j \in D} d_j \pi_j,$$

and since $c_{ij} = \bar{c}_{ij} + \pi_i - \pi_j$,

$$\sum_{i \in P} \sum_{j \in D} c_{ij} x_{ij} = \sum_{i \in P} \sum_{j \in D} \bar{c}_{ij} x_{ij} + \sum_{i \in P} a_i \pi_i - \sum_{j \in D} d_j \pi_j.$$

This is the Lemma «cost and reduced cost» of Chapter 4 for the bipartite network.

Definition 7.9: potentials of a basis

The **potentials of a basis** T are the values π_v , $v \in P \cup D$, such that

$$\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j = 0, \quad \forall (i, j) \in T, \quad \pi_{m+n} = 0.$$

- The potentials of a basis exist and are unique. Starting from $\pi_{m+n} = 0$, a basic cell (i, j) with one known endpoint gives the other one: $\pi_i = \pi_j + c_{ij}$ or $\pi_j = \pi_i - c_{ij}$. Each node is reached along its unique path in the tree from $m + n$, which forces its value.

Theorem 7.10: optimality criterion

Let $\bar{\mathbf{x}}$ be the basic feasible solution of a basis T , with potentials $\boldsymbol{\pi}$. If $\bar{c}_{ij} \geq 0$ for every cell, then $\bar{\mathbf{x}}$ is optimal for the TP model, $\boldsymbol{\pi}$ is optimal for the $D(\text{TP})$ model, and

$$\sum_{i \in P} \sum_{j \in D} c_{ij} \bar{x}_{ij} = \sum_{i \in P} a_i \pi_i - \sum_{j \in D} d_j \pi_j.$$

If moreover $\bar{c}_{ij} > 0$ for every cell outside T , then $\bar{\mathbf{x}}$ is the unique optimal solution.

Proof. We use the identity above. Let $z_\pi = \sum_{i \in P} a_i \pi_i - \sum_{j \in D} d_j \pi_j$.

1. For $\bar{\mathbf{x}}$: $\bar{c}_{ij} = 0$ on T and $\bar{x}_{ij} = 0$ outside T , so $\sum \bar{c}_{ij} \bar{x}_{ij} = 0$ and the cost of $\bar{\mathbf{x}}$ is z_π .
2. For every feasible $\mathbf{x}$: $\bar{c} \geq 0$ and $\mathbf{x} \geq \mathbf{0}$, so its cost is at least z_π ; hence $\bar{\mathbf{x}}$ is optimal. The potentials satisfy (7.2b), and their value z_π equals a primal cost: by weak duality they are optimal.
3. If $\bar{c}_{ij} > 0$ outside T , an optimal $\mathbf{x}$ has $\sum \bar{c}_{ij} x_{ij} = 0$, hence $x_{ij} = 0$ outside T : it is the basic solution of T , which is $\bar{\mathbf{x}}$ by [Theorem 7.6](#). $\square$

7.4.1 The cycle of a cell and the pivot

Lemma 7.11: the cycle of a non-basic cell

Let T be a basis and $e = (k, l) \notin T$. The cells $T \cup \{e\}$ contain exactly one cycle C , made of e and of the path from l to k in the tree. Write C starting from e and moving first along row k , and give the sign $+$ to the cells in the odd positions (e included) and $-$ to the others. Then every row and every column that meets C contains exactly one cell of C with sign $+$ and one with sign $-$.

Proof. The cycle is the fundamental cycle of the arc (k, l) (Proposition «fundamental cycle» of [Chapter 1](#)). It visits the nodes $k, j_1, i_1, j_2, \dots, l$ alternately in P and in D , so it has an even number of cells, and two consecutive cells share a node: the first two share the row k , the next two a column, and so on. Every node of C lies in exactly two cells of C , which are consecutive along C , the last and the first one included: their positions have different parity, hence different signs. $\square$

Theorem 7.12: the pivot

Let $\bar{\mathbf{x}}$ be the basic feasible solution of a basis T with potentials $\boldsymbol{\pi}$, let $e \notin T$ with $\bar{c}_e < 0$, let C^+ and C^- be the cells of its cycle with sign $+$ and $-$, and let

$$\theta = \min\{\bar{x}_f : f \in C^-\},$$

attained at a cell $f^* \in C^-$. For $0 \leq t \leq \theta$, let $\mathbf{x}(t)$ be $\bar{\mathbf{x}}$ with t added on the cells of C^+ and subtracted on the cells of C^- .

- (a) $\mathbf{x}(t)$ is feasible, and its cost is the cost of $\bar{\mathbf{x}}$ plus $t\bar{c}_e$.
- (b) $T' = T \cup \{e\} \setminus \{f^*\}$ is a basis, and $\mathbf{x}(\theta)$ is its basic feasible solution.

Proof. 1. *Feasibility.* By Lemma 7.11 every row and column of C gains t on one cell and loses t on another, so all the sums are unchanged. The cells of C^- keep $\bar{x}_f - t \geq 0$ since $t \leq \theta$.

2. *Cost.* The change of cost is $t(\sum_{C^+} c_{ij} - \sum_{C^-} c_{ij})$. Every node of C lies in one cell of C^+ and one of C^- , so $\sum_{C^+} (\pi_i - \pi_j) = \sum_{C^-} (\pi_i - \pi_j)$, and the change is $t(\sum_{C^+} \bar{c}_{ij} - \sum_{C^-} \bar{c}_{ij}) = t\bar{c}_e$: the other cells of C are basic, with $\bar{c}_{ij} = 0$.

3. *New basis.* T' has $m + n - 1$ cells, and it has no cycle: the only cycle of $T \cup \{e\}$ is C , which contains f^* . An acyclic set of $m + n - 1$ arcs on $m + n$ nodes is a spanning tree (Theorem «characterizations of a tree» of Chapter 1). $\mathbf{x}(\theta)$ is zero outside T' , since $x_{f^*}(\theta) = 0$, so it is the basic solution of T' . $\square$

Algorithm

Algorithm: The potential method

Input: a balanced instance with supplies a_i , demands d_j and costs c_{ij}

Output: an optimal basic feasible solution $\tilde{\mathbf{x}}$ and optimal potentials $\tilde{\pi}$

- 1 **Step 0 (initial basis).** Build a basis T and its basic feasible solution $\bar{\mathbf{x}}$ with the north-west corner rule or the minimum cost rule.
- 2 **Step 1 (potentials).** Compute the potentials of T and the reduced costs $\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j$ of the cells outside T .
- 3 **Step 2 (optimality test).** If $\bar{c}_{ij} \geq 0$ for every cell, STOP: $\bar{\mathbf{x}}$ is optimal.
- 4 **Step 3 (entering cell).** Choose a cell e with $\bar{c}_e < 0$.
- 5 **Step 4 (cycle).** Find the cycle of e with its signs, $\theta := \min\{\bar{x}_f : f \in C^-\}$, and a leaving cell $f^* \in C^-$ with $\bar{x}_{f^*} = \theta$.
- 6 **Step 5 (pivot).** Add θ on C^+ , subtract θ on C^- , set $T := T \cup \{e\} \setminus \{f^*\}$, and go to Step 1.

- **Rules of the examples.** In Step 3 the entering cell has the most negative reduced cost (the smallest i , then j , on ties). In Step 4 the cycle is written from the entering cell moving first along its row, and the leaving cell is the first cell of C^- with $\bar{x}_f = \theta$ in row-column order.
- The method is also known as the **u-v method** or MODI (modified distribution) method, and the cycle as the **stepping stone** path: the amounts move along it like on stones across a river.

Theorem 7.13: termination

If $\theta > 0$ at every iteration, the potential method stops after a finite number of iterations with an optimal basic feasible solution and optimal potentials.

Proof. By [Theorem 7.12](#) every iteration with $\theta > 0$ moves to a new basis and decreases the cost by $\theta |\bar{c}_e| > 0$. A basis determines its basic solution ([Theorem 7.6](#)), so a basis visited twice would have two different costs: no basis repeats. There are finitely many sets of $m + n - 1$ cells, so the method stops, at Step 2, and [Theorem 7.10](#) applies. $\square$

- When $\theta = 0$ the pivot is **degenerate**: the basis changes, the solution and its cost do not. The potential method is the simplex algorithm on the TP model: the basic cells are the basic variables, $\bar{c}_{ij}$ are their reduced costs, and the cycle describes how the basic variables change when x_e grows.
- Hence the anti-cycling rule of Bland of [Chapter 7 «Optimality conditions and pivot rules»](#) of the module Linear Programming applies: with the first cell with $\bar{c}_{ij} < 0$ in row–column order as the entering cell, and the leaving rule above, no basis repeats even with degenerate pivots.

The u-v convention

Many textbooks write the u-v method with $u_i + v_j = c_{ij}$ on the basic cells and $\bar{c}_{ij} = c_{ij} - u_i - v_j$. These are the potentials of this chapter with $u_i = \pi_i$ for the origins and $v_j = -\pi_j$ for the destinations: the reduced costs are the same, and the potentials of the destinations change sign.

7.4.2 The potential method on the transportation instance

- We start from the north-west corner solution of [Example 7.5](#), of cost 440.
- In each tableau the basic cells (blue) contain the amount $\bar{x}_{ij}$ in bold, and the other cells the reduced cost $\bar{c}_{ij}$; the last column and the last row contain the potentials. The entering cell is red, and the cells of the cycle are marked $\oplus$ and $\ominus$.

Example 7.7: the potential method from the north-west corner

In the first iteration the potentials of the north-west basis come from $\pi_7 = 0$, along the tree:

$$\pi_2 = \pi_7 + c_{27} = 2, \quad \pi_3 = \pi_7 + c_{37} = 6, \quad \pi_5 = \pi_2 - c_{25} = -2,$$

$$\pi_6 = \pi_2 - c_{26} = 0, \quad \pi_1 = \pi_5 + c_{15} = 7, \quad \pi_4 = \pi_1 - c_{14} = 6.$$

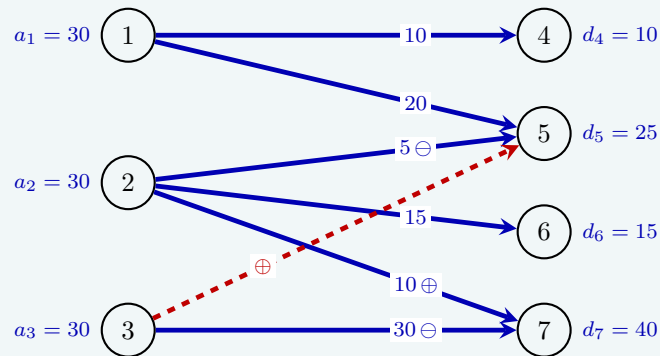
For example, $\bar{c}_{35} = c_{35} - \pi_3 + \pi_5 = 1 - 6 - 2 = -7$.

Iteration 1

	4	5	6	7	π_i
1	10	20	−5	−4	7
2	7	5 $\ominus$	15	10 $\oplus$	2
3	9	−7 $\oplus$	−2	30 $\ominus$	6
π_j	6	−2	0	0	

- The negative reduced costs are $\bar{c}_{35} = -7$, $\bar{c}_{16} = -5$, $\bar{c}_{17} = -4$, $\bar{c}_{36} = -2$; the cell $(3, 5)$ enters the basis.
- The cycle is $(3, 5)^+$, $(3, 7)^-$, $(2, 7)^+$, $(2, 5)^-$.
- The cells with sign $-$ have $\bar{x}_{37} = 30$ and $\bar{x}_{25} = 5$: $\theta = 5$, and the cell $(2, 5)$ leaves the basis.
- The cost goes from 440 to $440 + 5 \cdot (-7) = 405$.

In the network, the entering arc $(3, 5)$ (red, dashed) closes the cycle $3, 5, 2, 7, 3$ with the tree.



In the second iteration the cycle has six cells and visits all three rows.

Iteration 2

	4	5	6	7	π_i
1	10	20 $\ominus$	-12 $\oplus$	-11	14
2	14	7	15 $\ominus$	15 $\oplus$	2
3	16	5 $\oplus$	-2	25 $\ominus$	6
π_j	13	5	0	0	

- The negative reduced costs are $\bar{c}_{16} = -12$, $\bar{c}_{17} = -11$, $\bar{c}_{36} = -2$; the cell $(1, 6)$ enters the basis.
- The cycle is $(1, 6)^+$, $(1, 5)^-$, $(3, 5)^+$, $(3, 7)^-$, $(2, 7)^+$, $(2, 6)^-$.
- The cells with sign $-$ have $\bar{x}_{15} = 20$, $\bar{x}_{37} = 25$ and $\bar{x}_{26} = 15$: $\theta = 15$, and the cell $(2, 6)$ leaves the basis.
- The cost goes from 405 to $405 + 15 \cdot (-12) = 225$.

Iteration 3

	4	5	6	7	π_i
1	10	5 $\ominus$	15	-11 $\oplus$	14
2	14	7	12	30	2
3	16	20 $\oplus$	10	10 $\ominus$	6
π_j	13	5	12	0	

- The only negative reduced cost is $\bar{c}_{17} = -11$: the cell $(1,7)$ enters the basis.
- The cycle is $(1,7)^+$, $(1,5)^-$, $(3,5)^+$, $(3,7)^-$.
- The cells with sign $-$ have $\bar{x}_{15} = 5$ and $\bar{x}_{37} = 10$: $\theta = 5$, and the cell $(1,5)$ leaves the basis.
- The cost goes from 225 to $225 + 5 \cdot (-11) = 170$.

Iteration 4

	4	5	6	7	π_i
1	10	11	15 $\ominus$	5 $\oplus$	3
2	3	7	1	30	2
3	5	25	-1 $\oplus$	5 $\ominus$	6
π_j	2	5	1	0	

- The only negative reduced cost is $\bar{c}_{36} = -1$: the cell $(3,6)$ enters the basis.
- The cycle is $(3,6)^+$, $(3,7)^-$, $(1,7)^+$, $(1,6)^-$.
- The cells with sign $-$ have $\bar{x}_{37} = 5$ and $\bar{x}_{16} = 15$: $\theta = 5$, and the cell $(3,7)$ leaves the basis.
- The cost goes from 170 to $170 + 5 \cdot (-1) = 165$.

Iteration 5

	4	5	6	7	π_i
1	10	10	10	10	3
2	3	6	1	30	2
3	6	25	5	1	5
π_j	2	4	1	0	

- All the reduced costs are non-negative: the basic feasible solution is optimal, with cost $z = 165$.

- The optimal solution is

$$\tilde{x}_{14} = 10, \quad \tilde{x}_{16} = 10, \quad \tilde{x}_{17} = 10, \quad \tilde{x}_{27} = 30, \quad \tilde{x}_{35} = 25, \quad \tilde{x}_{36} = 5, \quad z(\text{TP}) = 165.$$

Every cell outside the basis has $\bar{c}_{ij} > 0$, so by [Theorem 7.10](#) the optimum is unique.

- The potentials $\tilde{\pi} = (3, 2, 5, 2, 4, 1, 0)$ are optimal for the $D(\text{TP})$ model, with value

$$30(3 + 2 + 5) - (10 \cdot 2 + 25 \cdot 4 + 15 \cdot 1 + 40 \cdot 0) = 300 - 135 = 165.$$

Gurobi returns the same optimal solution, and its dual values coincide with $\tilde{\pi}$.

- From the minimum cost solution of [Example 7.6](#) the method needs a single pivot: that solution has the basis of iteration 4, and the entering cell $(3, 6)$, with $\theta = 5$, gives the optimum.

7.4.3 Degeneracy

Example 7.8: a degenerate pivot

We consider the instance

c_{ij}	4	5	6	a_i
1	1	4	7	30
2	6	2	5	25
3	3	9	5	30
d_j	30	30	25	85

- In the north-west corner rule the first cell exhausts row 1 and column 4 at once. Only the row is crossed out, and the cell $(2, 4)$ enters T with the amount 0: the solution, of cost 250, is degenerate.

step	cell	amount	crossed out
1	(1, 4)	30	row 1
2	(2, 4)	0	column 4
3	(2, 5)	25	row 2
4	(3, 5)	5	column 5
5	(3, 6)	25	row 3, column 6

Iteration 1

	4	5	6	π_i
1	30	7	14	-7
2	0 $\ominus$	25 $\oplus$	7	-2
3	-10 $\oplus$	5 $\ominus$	25	5
π_j	-8	-4	0	

- The only negative reduced cost is $\bar{c}_{34} = -10$: the cell $(3, 4)$ enters the basis.
- The cycle is $(3, 4)^+$, $(3, 5)^-$, $(2, 5)^+$, $(2, 4)^-$.
- The cells with sign $-$ have $\bar{x}_{35} = 5$ and $\bar{x}_{24} = 0$: $\theta = 0$, and the cell $(2, 4)$ leaves the basis.
- The pivot is degenerate: the solution and its cost $z = 250$ do not change, only the basis does.

In the second iteration the basis has changed, so the potentials have changed too: now $\bar{c}_{15} < 0$.

Iteration 2

	4	5	6	π_i
1	30 $\ominus$	-3 $\oplus$	4	3
2	10	25	7	-2
3	0 $\oplus$	5 $\ominus$	25	5
π_j	2	-4	0	

- The only negative reduced cost is $\bar{c}_{15} = -3$: the cell $(1, 5)$ enters the basis.
- The cycle is $(1, 5)^+$, $(1, 4)^-$, $(3, 4)^+$, $(3, 5)^-$.
- The cells with sign $-$ have $\bar{x}_{14} = 30$ and $\bar{x}_{35} = 5$: $\theta = 5$, and the cell $(3, 5)$ leaves the basis.
- The cost goes from 250 to $250 + 5 \cdot (-3) = 235$.

Iteration 3

	4	5	6	π_i
1	25	5	4	3
2	7	25	4	1
3	5	3	25	5
π_j	2	-1	0	

- All the reduced costs are non-negative: the basic feasible solution is optimal, with cost $z = 235$.

The optimal solution is unique, with cost 235, and Gurobi confirms it.

- The basic cell with value 0 must be kept in T : without it, the other four cells would not span the six nodes, and the potentials could not be computed.
- The degenerate pivot of iteration 1 does not lower the cost, but it moves to a basis whose potentials reveal an improving cell.

The potential method computes the potentials of the basis ($\bar{c}_{ij} = 0$ on the basic cells, $\pi_{m+n} = 0$), stops when every reduced cost is non-negative, and otherwise moves θ units around the cycle of an entering cell. The cost decreases by $\theta |\bar{c}_e|$; with $\theta = 0$ only the basis changes.

7.5 The assignment problem

Definition 7.14: assignment problem

There are n workers, the origins $P = \{1, 2, \dots, n\}$, and n jobs, the destinations $D = \{n+1, n+2, \dots, 2n\}$; c_{ij} is the cost of assigning worker i to job j . An **assignment** gives every worker exactly one job and every job to exactly one worker. The **assignment problem** asks for an assignment of minimum total cost.

- We use one binary variable x_{ij} for every pair $i \in P, j \in D$, equal to 1 if worker i gets job j .

Using these variables, an Integer Linear Programming (ILP) model for the assignment problem reads as follows:

$$\min \sum_{i \in P} \sum_{j \in D} c_{ij} x_{ij} \quad (7.3a)$$

$$\text{subject to } \sum_{j \in D} x_{ij} = 1, \quad \forall i \in P, \quad (7.3b)$$

$$\sum_{i \in P} x_{ij} = 1, \quad \forall j \in D, \quad (7.3c)$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in P, \forall j \in D. \quad (7.3d)$$

This model is referred to as the AP model.

- Description of the objective function and constraints of the AP model:
 - the objective function (7.3a) minimizes the total cost of the assignment;
 - constraints (7.3b) give every worker exactly one job;

- constraints (7.3c) give every job to exactly one worker;
- finally, constraints (7.3d) define the binary variables of the model.
- An $n \times n$ matrix with entries in $\{0, 1\}$ and exactly one 1 in every row and in every column is a **permutation matrix**. The feasible solutions of the AP model are the permutation matrices: there are $n!$ assignments.

Proposition 7.15: the LP relaxation of the assignment model

The LP relaxation of the AP model, with $x_{ij} \geq 0$ in place of (7.3d), is the TP model with $a_i = 1$ and $d_j = 1$. Its vertices are permutation matrices, and it has an optimal solution that is an assignment.

Proof. The constraints (7.3b) and (7.3c) are (7.1b) and (7.1c) with unit supplies and demands, and the instance is balanced, with total n . By Proposition 7.2 it has an optimal solution at a vertex, and every vertex is integer by the Theorem «integer vertices» of Chapter 4. An integer feasible solution has $0 \leq x_{ij} \leq 1$, since x_{ij} is at most the sum of its row: it is a permutation matrix. $\square$

Example 7.9: four workers and four jobs

Four workers 1, 2, 3, 4 and four jobs 5, 6, 7, 8 have the costs

c_{ij}	5	6	7	8
1	2	4	3	5
2	4	8	1	1
3	2	6	9	6
4	1	9	8	6

The $4! = 24$ assignments, enumerated, have a unique minimum: worker 1 gets job 7, worker 2 job 8, worker 3 job 6, worker 4 job 5, with cost $3 + 1 + 6 + 1 = 11$. Gurobi gives the same assignment as the optimal solution of the LP relaxation.

7.5.1 Degeneracy of the potential method

- A basis of the LP relaxation has $2n - 1$ cells, but an assignment has only n cells with $x_{ij} = 1$: every basic feasible solution has $n - 1$ basic cells with value 0, and it is degenerate.
- The potential method solves the LP relaxation, but many of its pivots have $\theta = 0$. On [Example 7.9](#) the north-west corner rule gives the basis below, with three basic zeros; from it the potential method reaches the optimum after 6 pivots, 4 of them degenerate.

$\bar{x}_{ij}$	5	6	7	8	a_i
1	1				1
2	0	1			1
3			0	1	1
4				0	1
d_j	1	1	1	1	

7.5.2 Reductions and potentials

Lemma 7.16: reduced costs of an assignment

For potentials π , let $\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j$. For every $\mathbf{x}$ that satisfies (7.3b) and (7.3c),

$$\sum_{i \in P} \sum_{j \in D} c_{ij} x_{ij} = \sum_{i \in P} \sum_{j \in D} \bar{c}_{ij} x_{ij} + \sum_{i \in P} \pi_i - \sum_{j \in D} \pi_j.$$

Hence subtracting a constant from a row or from a column of the cost matrix changes the cost of every assignment by the same constant, and the optimal assignments do not change.

Proof. It is the identity of [Section 7.4](#) with $a_i = d_j = 1$. Subtracting r from row i gives the reduced costs of $\pi_i = r$ and the other potentials 0; subtracting s from column j those of $\pi_j = -s$. $\square$

Corollary 7.17: zero-cost assignments are optimal

If $\bar{c}_{ij} \geq 0$ for every cell and an assignment uses only cells with $\bar{c}_{ij} = 0$, then it is optimal, even for the LP relaxation, its cost is $\sum_{i \in P} \pi_i - \sum_{j \in D} \pi_j$, and π is optimal for the dual of the LP relaxation, the $D(\text{TP})$ model with $a_i = d_j = 1$.

Proof. By [Lemma 7.16](#), the assignment costs $\sum_{i \in P} \pi_i - \sum_{j \in D} \pi_j$, and every $\mathbf{x} \geq \mathbf{0}$ satisfying (7.3b) and (7.3c) costs at least as much, since $\bar{c} \geq 0$. The potentials satisfy $\pi_i - \pi_j \leq c_{ij}$ with the same value, so they are optimal by weak duality. $\square$

7.5.3 Independent zeros and lines

Definition 7.18: independent zeros and covers

Let $\bar{c}$ be an $n \times n$ matrix with non-negative entries. A set of zero entries is **independent** if no two of them lie in the same row or in the same column. A set of **lines**, rows and columns, **covers** the zeros if every zero entry lies in at least one of the lines.

- n independent zeros form an assignment of reduced cost zero, optimal by [Corollary 7.17](#).
- In the bipartite network whose arcs are the cells with $\bar{c}_{ij} = 0$, a set of independent zeros is a matching, a set of arcs without common endpoints as in [Chapter 5 «Maximum flow»](#), and a cover is a set of nodes that touches every arc.

Theorem 7.19: König and Egerváry

The maximum number of independent zeros of a matrix equals the minimum number of lines that cover all its zeros.

Proof. Let M be a maximum set of independent zeros. A row or a column is **matched** if it contains a zero of M .

1. *Lines $\geq$ zeros.* The zeros of M lie in different rows and columns, so a line contains at most one of them, and a cover needs at least $|M|$ lines.
2. *Alternating paths.* An **alternating path** starts at an unmatched row, goes from a row to a column through a zero not in M , and from a column to a row through the zero of M of that column. Let R and K be the rows and the columns reached by alternating paths, the unmatched rows included.
3. *Every column of K is matched.* Otherwise an alternating path ends at an unmatched column. Along it, the zeros not in M are one more than the zeros of M : exchanging them gives an independent set with $|M| + 1$ zeros, against the maximality of M .
4. *The lines.* Let L be the rows not in R and the columns in K . A zero (i, j) with row $i \in R$ has column $j \in K$: if it is not in M , a path to i continues to j ; if it is in M , row i is matched, so it was reached from its column j . Hence L covers all the zeros.
5. *Counting.* The rows not in R are matched, since the unmatched rows are in R , and the columns of K are matched. A zero of M in a column of K has its row in R , so it lies on exactly one line of L . Lines and zeros of M correspond one to one, and $|L| = |M|$. □

- The proof is constructive: it gives the lines of a minimum cover from a maximum set of independent zeros, and a set that is not maximum is enlarged along an alternating path ending at an unmatched column.

7.5.4 The Hungarian algorithm

Algorithm

Algorithm: The Hungarian algorithm

Input: an $n \times n$ cost matrix c_{ij} , $i \in P$, $j \in D$

Output: an optimal assignment and optimal potentials π

- 1 **Step 0 (reduction).** Subtract from each row its minimum, π_i ; then subtract from each column of the result its minimum, $-\pi_j$. Let $\bar{c}$ be the reduced matrix.
- 2 **Step 1 (independent zeros).** Find a maximum set M of independent zeros of $\bar{c}$.
- 3 If $|M| = n$, STOP: M is an optimal assignment.
- 4 **Step 2 (lines).** Cover the zeros with $|M| < n$ lines, as in the proof of the theorem of König and Egerváry.
- 5 **Step 3 (adjustment).** Let δ be the minimum uncovered entry. Subtract δ from the uncovered entries and add it to the entries covered twice: $\pi_i := \pi_i + \delta$ for every uncovered row, $\pi_j := \pi_j + \delta$ for every covered column. Go to Step 1.

- **Rule of the examples.** In Step 1 the set M is built row by row, in increasing order: each row looks for an alternating path to an unmatched column, depth first, trying the columns in increasing order.
- The algorithm is due to H. W. Kuhn (1955), who named it after the Hungarian mathematicians D. König and J. Egerváry.

Theorem 7.20: correctness of the Hungarian algorithm

At every iteration $\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j \geq 0$ for every cell, and Step 3 increases the dual value $\sum_{i \in P} \pi_i - \sum_{j \in D} \pi_j$ by $\delta(n - k) > 0$, where $k < n$ is the number of lines. With integer costs the algorithm stops after a finite number of iterations with an optimal assignment, and π is optimal for the dual of the LP relaxation.

- Proof.*
1. *Step 0.* Subtracting the minimum of a row leaves non-negative entries and a zero in the row; the column minima are then ≥ 0 , and subtracting them keeps the entries non-negative. By construction $\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j$.
 2. *Step 3.* All zeros are covered, so $\delta > 0$, and the uncovered entries stay ≥ 0 . The update of the potentials gives $-\delta$ on uncovered row and column, $-\delta + \delta = 0$ on uncovered row and covered column, $+\delta$ on covered row and column, 0 on covered row and uncovered column: exactly the adjustment.
 3. *Dual value.* With k_r covered rows and k_c covered columns, the value grows by $\delta(n - k_r) - \delta k_c = \delta(n - k)$, and $k = |M| < n$ by [Theorem 7.19](#).
 4. *Termination.* By [Lemma 7.16](#) the dual value never exceeds the cost of an assignment. With integer costs every δ is an integer ≥ 1 , so the number of adjustments is finite. At the stop M is an assignment of zeros, optimal with π by [Corollary 7.17](#). □

7.5.5 The Hungarian algorithm on the example

Example 7.10: the Hungarian algorithm

We run the algorithm on [Example 7.9](#).

Step 0. The row minima are 2, 1, 2, 1, so $\pi_1 = 2$, $\pi_2 = 1$, $\pi_3 = 2$, $\pi_4 = 1$, and the rows become

$c_{ij} - \pi_i$	5	6	7	8
1	0	2	1	3
2	3	7	0	0
3	0	4	7	4
4	0	8	7	5

The column minima are 0, 2, 0, 0: only column 6 changes, and $\pi_6 = -2$. The dual value $2 + 1 + 2 + 1 + 2 = 8$ is a lower bound on the cost of every assignment.

In each of the following matrices the independent zeros are boxed, the lines blue and the cell of δ red; the potentials are on the margins.

Iteration 1

$\bar{c}_{ij}$	5	6	7	8	π_i
1	0	0	1	3	2
2	3	5	0	0	1
3	0	2	7	4	2
4	0	6	7	5	1
π_j	0	-2	0	0	

- A maximum set of independent zeros (boxed) has 3 elements: (1, 6), (2, 7), (3, 5).
- The $3 < 4$ lines (blue) are the rows 1 and 2 and the column 5.
- The minimum uncovered entry is $\delta = 2$, in the cell (3, 6) (red). We subtract it from the uncovered entries and add it to the entries covered twice: π_i grows by 2 for the uncovered rows 3 and 4 and π_j grows by 2 for the covered column 5. The dual value grows by $2 \cdot (4 - 3) = 2$.

Iteration 2

$\bar{c}_{ij}$	5	6	7	8	π_i
1	2	0	1	3	2
2	5	5	0	0	1
3	0	0	5	2	4
4	0	4	5	3	3
π_j	2	-2	0	0	

- A maximum set of independent zeros (boxed) has 3 elements: $(1, 6)$, $(2, 7)$, $(3, 5)$.
- The $3 < 4$ lines (blue) are the row 2 and the columns 5 and 6.
- The minimum uncovered entry is $\delta = 1$, in the cell $(1, 7)$ (red). We subtract it from the uncovered entries and add it to the entries covered twice: π_i grows by 1 for the uncovered rows 1, 3 and 4 and π_j grows by 1 for the covered columns 5 and 6. The dual value grows by $1 \cdot (4 - 3) = 1$.

Iteration 3					
$\bar{c}_{ij}$	5	6	7	8	π_i
1	2	0	0	2	3
2	6	6	0	0	1
3	0	0	4	1	5
4	0	4	4	2	4
π_j	3	-1	0	0	

- A maximum set of independent zeros (boxed) has 4 elements: $(1, 7)$, $(2, 8)$, $(3, 6)$, $(4, 5)$.
- They form an assignment of zero reduced cost: the algorithm stops, and the assignment is optimal.

- The optimal assignment is $1 \rightarrow 7$, $2 \rightarrow 8$, $3 \rightarrow 6$, $4 \rightarrow 5$, with cost $3 + 1 + 6 + 1 = 11$, the unique optimum of [Example 7.9](#).
- The final potentials $\pi_1 = 3$, $\pi_2 = 1$, $\pi_3 = 5$, $\pi_4 = 4$ and $\pi_5 = 3$, $\pi_6 = -1$, $\pi_7 = 0$, $\pi_8 = 0$ are an optimal dual solution, of value $(3 + 1 + 5 + 4) - (3 - 1 + 0 + 0) = 11$. The dual value grew from 8 to 10 and then to 11.
- The lines are those of the proof of [Theorem 7.19](#). In iteration 1 row 4 is unmatched; from it an alternating path reaches column 5 through its zero, and then row 3, matched to column 5. Hence $R = \{3, 4\}$, $K = \{5\}$, and the lines are the rows 1, 2 and the column 5.

The assignment problem is a transportation problem with unit supplies and demands, on which the potential method makes many degenerate pivots. The Hungarian algorithm keeps $\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j \geq 0$, covers the zeros with the minimum number of lines (König–Egerváry), and raises the dual value until n independent zeros give an optimal assignment.

7.6 Typical mistakes

Warning

- **Solving an unbalanced instance as it is.** With more supply than demand the equality constraints have no solution: add a dummy destination with zero costs, or a dummy origin when the demand exceeds the supply.
- **Dropping a basic zero.** A degenerate basis still has $m + n - 1$ cells; without the cell of value 0 the tree is disconnected and the potentials cannot be computed.
- **Taking a set with a cycle as a basis.** $m + n - 1$ cells are a basis only if they contain no closed path alternating rows and columns.
- **Mixing the sign conventions.** Here $\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j$ with $\pi_i - \pi_j = c_{ij}$ on the basic cells; with $u_i + v_j = c_{ij}$ the destination potentials change sign.
- **Computing θ on all the cells of the cycle.** θ is the minimum on the cells with sign $-$ only; the entering cell receives θ .
- **Covering the zeros with too many lines.** The lines of the Hungarian algorithm must be a minimum cover, as many as the independent zeros; with more lines δ may be wrong or the stop missed.
- **Forgetting the entries covered twice.** In Step 3 of the Hungarian algorithm δ is subtracted from the uncovered entries and added to those covered by a row and a column.
- **Running the algorithms on profits.** They minimize: with profits p_{ij} to maximize, use the costs $c_{ij} = -p_{ij}$, or $c_{ij} = p_{\max} - p_{ij}$ with $p_{\max}$ the largest profit, to keep them non-negative.

7.7 In summary

Summary

- The TP model ships the supplies a_i of m origins to the demands d_j of n destinations; it is an MCF model on the complete bipartite network, feasible exactly when it is balanced.
- An uncapacitated transshipment instance becomes a transportation instance with the distances as costs.
- A basis is a spanning tree of the bipartite network, with $m + n - 1$ cells; its basic solution is found leaf by leaf and is integer for integer data.
- The north-west corner and the minimum cost rules build a basic feasible solution, one line per step.
- The potential method: potentials with $\bar{c}_{ij} = 0$ on the basis and $\pi_{m+n} = 0$; optimal

when $\bar{c} \geq 0$; otherwise θ units move around the cycle of the entering cell, and the cost drops by $\theta |\bar{c}_e|$.

- The AP model is a TP model with unit data; the Hungarian algorithm reduces rows and columns, covers the zeros with a minimum number of lines (König–Egerváry) and adjusts by δ , raising the dual value.

The instances at a glance.

warehouses and stores	north-west 440, minimum cost 170, optimum 165
potential method	4 pivots from north-west, 1 from minimum cost
optimal potentials	$\tilde{\pi} = (3, 2, 5, 2, 4, 1, 0)$, with $\pi_7 = 0$
with $a_1 = 40$	dummy destination 8, optimum 155
degenerate instance	$\theta = 0$, then $\theta = 5$; optimum 235
transshipment, no capacities	optimum 49
workers and jobs	bound 8, $\delta = 2$, $\delta = 1$, optimum 11