Operations research meets data science: principles, models and algorithms

Immanuel Bomze, University of Vienna

Dip. Matematica@Univ. Padova 06 November 2025

Example, continued

... with, say, $\mathbf{c}^\top = [-1, 2, 3 \,|\, 0, 0, 0, 0]$ (recall: last four are slacks)

$$\mathsf{A} = \begin{bmatrix} 1 & 1 & 1 & | & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & | & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & | & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & | & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 4 \\ 2 \\ 3 \\ 6 \end{bmatrix}.$$

Start with vertex $\mathbf{z}_{old} = [\mathbf{o}^\top | \mathbf{s}_{old}^\top]$, so here $\mathbf{s}_{old} = \mathbf{b}$ and $\mathbf{c}_\mathsf{B} = \mathbf{o}$.

Calculate $\mathbf{c}^{red} = (\mathbf{c}_\mathsf{N}^\top - \mathbf{c}_\mathsf{B}^\top \mathsf{B}^{-1}\mathsf{N})^\top = \mathbf{c}_\mathsf{N} = [-1, 2, 3]^\top \not\geq \mathbf{o}$.

Hence $\mathbf{z}_{old}$ not optimal, pass to next vertex with $z_1^{new} > 0$

Only such vertex is $\mathbf{z}_{new}^\top = [2, 0, 0 \,|\, 2, 0, 3, 6]$. Compare with previous vertex: $\mathbf{z}_{old}^\top = [0, 0, 0 \,|\, 4, 2, 3, 6]$.

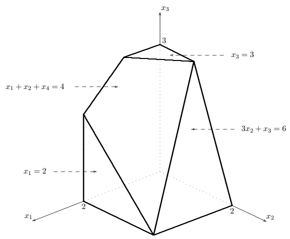
New reduced cost vector is $\mathbf{c}_{new}^{red} = [1, 2, 3]^\top \geq \mathbf{o}$, so **stop:**

$\mathbf{x}^* = [2, 0, 0]^\top$ is optimal (first part of $\mathbf{z}_{new}$).

Simplex Method for LP – ideas, but not yet an algorithm !

Why ? Still imprecise/incomplete:

Which neighbor vertex ?



How to detect unboundedness ?

$x_1 - x_2 \leq 2, \; \mathbf{x} \in \mathbb{R}^2_+$

How to start ? How to detect infeasibility ?

Last issue leads to *Phase I* of Simplex Method.

Feasibility of an LP – Phase I

Assume LP already in standard form, min $\left\{\mathbf{c}^\top\mathbf{x} : \mathsf{A}\mathbf{x} = \mathbf{b}\,,\ \mathbf{x} \in \mathbb{R}^n_+\right\}$.

Assume A has full row rank m and that $\mathbf{b} \geq \mathbf{o}$
(multiply some equations by (-1), drop superfluous rows).
$\mathbf{x} = \mathbf{o}$ only feasible if $\mathbf{b} = \mathbf{o}$ but if $\mathbf{b} \neq \mathbf{o}$, use slacks:

$$[\,\mathsf{A}\,|\,\mathsf{I}_m\,]\mathbf{z} = \mathsf{A}\mathbf{x} + \mathbf{s} = \mathbf{b} \quad \text{with } \mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \mathbf{s} \end{bmatrix} \in \mathbb{R}^{m+n}_+$$

has always a solution $\mathbf{z}$ with $\mathbf{x} = \mathbf{o}$ and $\mathbf{s} = \mathbf{b} \in \mathbb{R}^m_+$.

So consider auxiliary LP min $\left\{\sum_i s_i : \mathsf{A}\mathbf{x} + \mathbf{s} = \mathbf{b}\,, \mathbf{z} \in \mathbb{R}^{m+n}_+\right\}$
in $\mathbf{z}$ which always is feasible and bounded (objective ≥ 0).

Solving Phase I of an LP

After finitely many vertex exchanges, arrive at optimal solution $\mathbf{z}^*$ of auxiliary LP;

either $\sum_i s_i^* > 0$, then original LP is infeasible;

or $\sum_i s_i^* = 0$, then $\mathbf{s}^* = \mathbf{o}$, thus $A\mathbf{x}^*[+\mathbf{o}] = \mathbf{b}$,

$\mathbf{x}^*$ is feasible vertex(!) of original LP.

Ex.: min $\left\{\mathbf{c}^\top \mathbf{x} : -x_1 - x_2 = 1\,,\, \mathbf{x} \in \mathbb{R}^2_+\right\}$ is obviously infeasible.

Auxiliary LP is min $\left\{s_1 : -x_1 - x_2 + s_1 = 1\,,\, [x_1, x_2, s_1]^\top \in \mathbb{R}^3_+\right\}$ has optimal solution $s_1^* = 1 > 0$, attained at starting vertex $\mathbf{z}^* = [0, 0, \mathbf{b} = 1]^\top$, as $\mathbf{c}^{red} = [1, 1]^\top \geq \mathbf{o}$.

Solving Phase I of a feasible LP

Ex. $M = \left\{\mathbf{x} \in \mathbb{R}^2_+ : x_1 + 2x_2 = 2\right\} \neq \emptyset$ is feasible but $\mathbf{o} \notin M$.

The auxiliary LP is constrained by $x_1 + 2x_2 + s_1 = 2$, starts at vertex $\mathbf{z}^{old} = [0, 0, 2]^\top$, with reduced cost $\mathbf{c}^{red} = [-1, -2]^\top$.

Choosing greedily $c_2^{red} = -2$, one vertex exchange gives $\mathbf{z}^{new} = [0, 1, 0]^\top$ which is optimal (to aux.LP) with $\mathbf{c}^{red}_{new} = [1, 1]^\top \geq \mathbf{o}$ and $s_1^{new} = 0$, so $\mathbf{x}^{new} = [0, 1]^\top$ is a vertex of M as desired, with which we can start the Phase II of the original LP.

Choosing instead $c_1^{red} = -1$ before vertex exchange would give $\mathbf{z}^{new} = [2, 0, 0]^\top$, again optimal to aux.LP, but now yielding $\mathbf{x}^{new} = [2, 0]^\top \in M$, a different starting vertex for Phase II.

A more complete algorithmic scheme of Simplex Method

Input: Problem data $(\mathsf{A}, \mathbf{b}, \mathbf{c})$, no redundant equations, $\mathbf{b} \geq \mathbf{o}$.

Phase I.: Deciding (in-)feasibility, determine starting vertex: solve aux.LP

$$\min \left\{ \sum_j s_j : \mathsf{A}\mathbf{x} + \mathbf{s} = \mathbf{b}\,,\ (\mathbf{x}, \mathbf{s}) \in \mathbb{R}^n_+ \times \mathbb{R}^m_+ \right\}$$

by application of Phase II below, starting with $\mathbf{x} = \mathbf{o}$, $\mathbf{s} = \mathbf{b}$.

Phase II.: Optimizing given LP, starting from given vertex:

Repeat until ($\mathbf{c}^{red} \geq \mathbf{o}$ or unboundedness certificate is met)

replace vertex $\mathbf{z}^{old}$ with improving neighbor vertex $\mathbf{z}^{new}$;

update $\mathbf{c}^{red}$ and decomposition $\mathsf{A} = [\,\mathsf{N}\,|\,\mathsf{B}\,]$.

Still some details missing ...

Transition Phase I $\rightarrow$ Phase II:

end vertex of Phase I (opt.solution to aux.LP) becomes starting vertex of Phase II.

Unboundedness certificate:

check signs of column i in $\mathsf{B}^{-1}\mathsf{N}$ if $c_i^{red} < 0$ selected.

Details of vertex exchange/update of [N | B] in Phase II.

Bottom line: choose m columns (out of n) of $m \times n$ matrix A and modify choice until optimal solution found.

Example, reconsidered

$$\min \left\{ \mathbf{c}^\top \mathbf{z} : \mathsf{A}\mathbf{z} = \mathbf{b}\,,\ \mathbf{z} \in \mathbb{R}^7_+ \right\}$$

... with, say, $\mathbf{c}^\top = [-1, 2, 3 \,|\, 0, 0, 0, 0]$

$$\mathsf{A} = \begin{bmatrix} 1 & 1 & 1 & | & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & | & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & | & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & | & 0 & 0 & 0 & 1 \end{bmatrix}, \ \mathbf{b} = \begin{bmatrix} 4 \\ 2 \\ 3 \\ 6 \end{bmatrix}.$$

Start with vertex $\quad \mathbf{z}_{old}^\top = [0, 0, 0 \,|\, 4, 2, 3, 6].$

Pass to next vertex $\quad \mathbf{z}_{new}^\top = [2, 0, 0 \,|\, 2, 0, 3, 6].$

Bottom line: choose 4 columns of A and ...

Features and performance of Simplex Method

Features: sparsity is preserved (only m variables > 0), cost reduction at every step, finite (!) procedure.

Average case: polynomial in data size (in practice $\leq (m+n)^3$ exchanges), effort per exchange step $\approx n*(m+n)$.

Worst case: exponential (all vertices visited).

From linear to nonlinear problems (NLP)

Formally, not much changes:

$$\min\{f(\mathbf{x}) : \mathbf{g}(\mathbf{x}) \le \mathbf{o}\,,\ \mathbf{h}(\mathbf{x}) = \mathbf{o}\,,\ \mathbf{x} \in X\}$$

where X is the *ground set* of simple structure ($X = \mathbb{R}^n, \mathbb{R}^n_+$, ...)

Ex.: LP in standard form recovered:

$$f(\mathbf{x}) = \mathbf{c}^\top\mathbf{x}\,,\ \mathbf{h}(\mathbf{x}) = \mathbf{b} - \mathsf{A}\mathbf{x}\,,\ X = \mathbb{R}^n_+\,.$$

Hope for finite/efficient algorithm ? Generally in vain ...

Ex.: $f(x) = (x-1)^2$, $X = \mathbb{R}^1_+$, $g(x) = x - 2$.
Feasible set $M = [0;2]$ with vertices $\{0,2\}$,
but optimal solution $x^* = 1$ in interior of M.

Can be anywhere in M for NLP.

Concave minimization solved at vertices

However, same proof as for LP works if f is *concave* function:

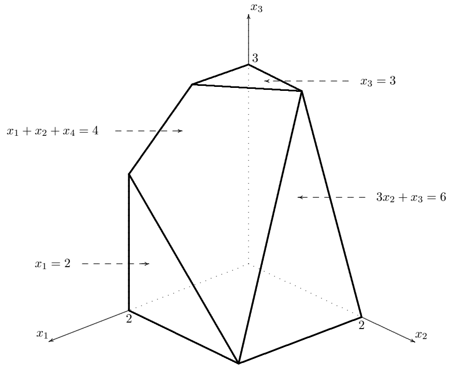
$$f(\tfrac{1}{2}\mathbf{x} + \tfrac{1}{2}\mathbf{y}) \geq \tfrac{1}{2}f(\mathbf{x}) + \tfrac{1}{2}f(\mathbf{y})$$

and if M is a convex set ($\frac{1}{2}\mathbf{x} + \frac{1}{2}\mathbf{y} \in M$ if $\{\mathbf{x}, \mathbf{y}\} \subset M$):

then optimal solution attained at extreme point/vertex of M.
Why ?

Even then no efficient way to check vertices —

suboptimal vertices may have
no improving neighbors.



Need something else for NLP ...

.. depending on structure:

- zero-order methods (derivative-free search)
- first-order methods (use gradients or substitutes of them)
- higher-order methods (use, e.g., curvature information)

Principle of line search

Unconstrained case min $\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^n\}$.

Repeat until stop criterion is met

1) at an iterate $\mathbf{x}^{old}$, decide a search direction $\mathbf{d} \in \mathbb{R}^n$;
2) search for optimal step length $t^* > 0$ by looking at

   min $\left\{f(\mathbf{x}^{old} + t\mathbf{d}) : t \geq 0\right\}$ — one-dimensional search;
3) update $\mathbf{x}^{new} = \mathbf{x}^{old} + t^*\mathbf{d}$ if $f(\mathbf{x}^{new}) < f(\mathbf{x}^{old})$.

output last $\mathbf{x}^{new}$ as (tentatively optimal) solution.

Constrained case: similar, restrictions on $\mathbf{d}$ and step size t.

Rigorous bounds – Lagrange duality

Again, general constrained optimization problem

$$\min\{f(\mathbf{x}) : \mathbf{g}(\mathbf{x}) \le \mathbf{o}\,,\, \mathbf{h}(\mathbf{x}) = \mathbf{o}\,,\, \mathbf{x} \in X\}$$

where $X\,(=\mathbb{R}^n, \mathbb{R}^n_+$, a grid) is ground set of simple structure.

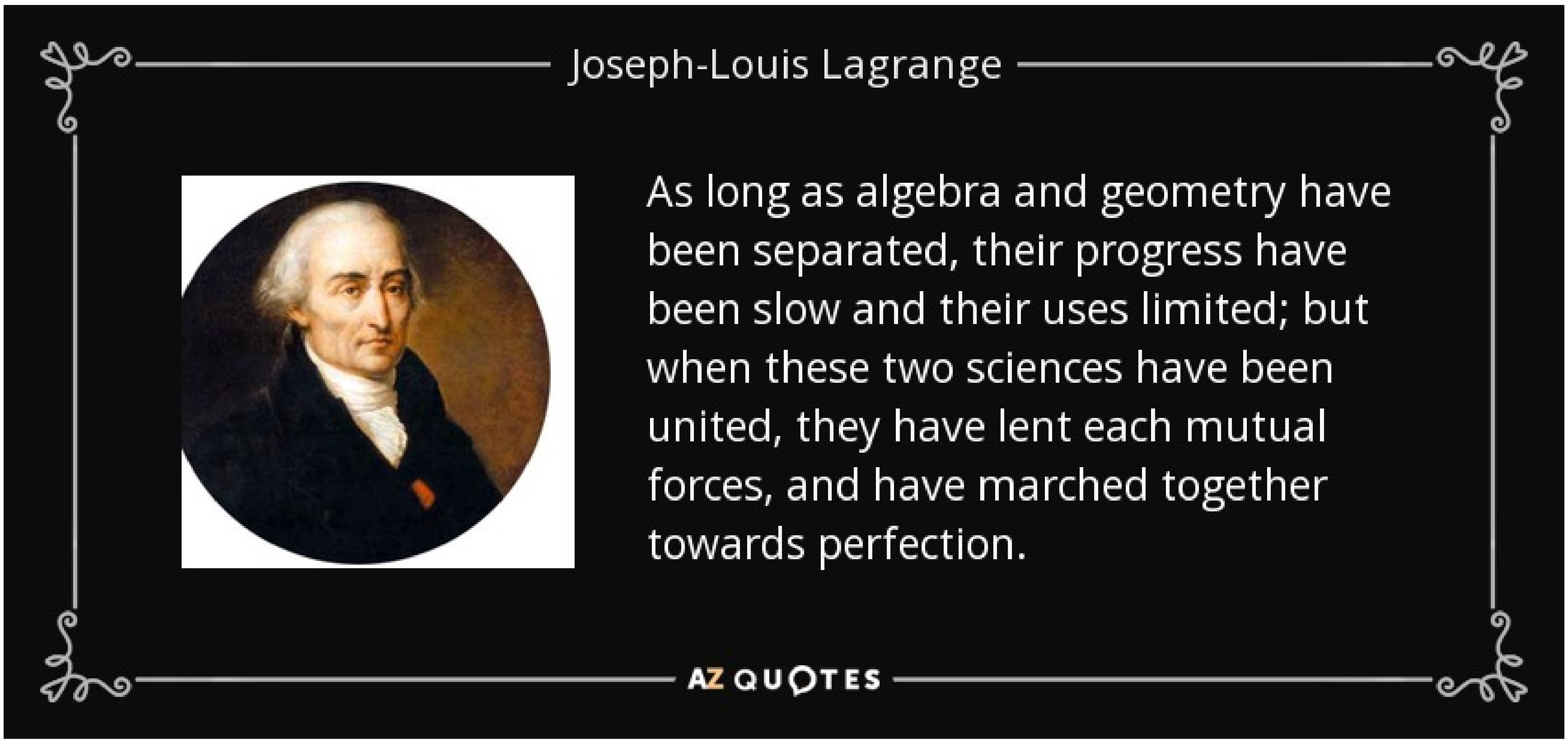
Assume have stopped at supposedly good solution $\mathbf{x}^*$,
but without optimality certificate.

Lagrange duality: idea of pricing constraints

$$L(\mathbf{x};\mathbf{y}) := f(\mathbf{x}) + \sum_{i=1}^{m} p_i g_i(\mathbf{x}) + \sum_{j=1}^{q} v_j h_j(\mathbf{x})$$

with $\mathbf{y}^\top = [\mathbf{p}^\top \,|\, \mathbf{v}^\top] \in \mathbb{R}^m_+ \times \mathbb{R}^q$ the multipliers (*dual variables*).

Joseph-Louis Lagrange aka Giuseppe Lodovico Lagrangia (Turin 1736 – Paris 1813)



Lagrange dual function gives lower bound

Observe: for any $\mathbf{x}$ feasible have $L(\mathbf{x};\mathbf{y}) \leq f(\mathbf{x})$, all $\mathbf{y} \in \mathbb{R}^m_+ \times \mathbb{R}^q$. Indeed $\mathbf{x} \in M$ entails $\mathbf{g}(\mathbf{x}) \leq \mathbf{o}$ and $\mathbf{h}(\mathbf{x}) = \mathbf{o}$, so

$$L(\mathbf{x};\mathbf{y}) = f(\mathbf{x}) + \sum_{i=1}^{m} p_i g_i(\mathbf{x}) + \sum_{j=1}^{q} v_j h_j(\mathbf{x}) \leq f(\mathbf{x}) \, .$$

Now minimize L instead of f in $\mathbf{x}$, but unconstrained:

$$\Theta(\mathbf{y}) := \min_{\mathbf{x} \in X} L(\mathbf{x};\mathbf{y})$$

... easier but need to solve for all $\mathbf{y}$.

Observe: for all $\mathbf{y} \in \mathbb{R}^m_+ \times \mathbb{R}^q$ have (why ?)

$$\Theta(\mathbf{y}) \leq \min_{\mathbf{x} \in M} L(\mathbf{x};\mathbf{y}) \leq \min_{\mathbf{x} \in M} f(\mathbf{x})$$

so all Θ values are a *lower bound* of optimal solution value.

Tightest lower bound — weak duality

Tightest lower bound: the *dual optimal solution value*

$$\max_{\mathbf{y}} \Theta(\mathbf{y}) \leq \min_{\mathbf{x} \in M} f(\mathbf{x}) \, .$$

This inequality is called *weak duality.*

Application: suppose have found $\mathbf{y}^* \in \mathbb{R}^m_+ \times \mathbb{R}^q$ such that

$$[\, 0 \leq \,] \; f(\mathbf{x}^*) - \Theta(\mathbf{y}^*) \leq 10^{-7} \, .$$

Then know that $\mathbf{x}^*$ is optimal up to 10^{-7}.

May stop with this almost optimality certificate.

Summarizing weak duality; dual problem

For any *primal-dual feasible pair* $(\mathbf{x}^*, \mathbf{y}^*)$, $\mathbf{x}^* \in M$, $\mathbf{y}^* \in \mathbb{R}^m_+ \times \mathbb{R}^q$, have

$$\Theta(\mathbf{y}^*) \leq \max_{\mathbf{y}} \Theta(\mathbf{y}) \leq \min_{\mathbf{x} \in M} f(\mathbf{x}) \leq f(\mathbf{x}^*)\,,$$

and if values at both ends of above inequality chain are close to each other, both almost optimal.

Trouble: how to find tightest $\Theta(\mathbf{y}^*)$ – *dual optimization problem*.

Hopeless ? Not quite if we have some structure ...

LP duality

Simplest structure: linear optimization (LP):

$$\left.\begin{array}{rcl} \mathbf{c}^\top \mathbf{x} & \to & \text{min !} \\ A\mathbf{x} & \geq & \mathbf{b} \\ \mathbf{x} & \geq & \mathbf{o} \end{array}\right\} \qquad \text{and} \qquad \left\{\begin{array}{rcl} \mathbf{b}^\top \mathbf{y} & \to & \text{max !} \\ A^\top \mathbf{y} & \leq & \mathbf{c} \\ \mathbf{y} & \geq & \mathbf{o} \end{array}\right.$$

are the dual optimization problems of each other.

Optimal values of both coincide – *duality gap* is zero.

Very powerful.

Duality in NLP – convex case

Nonlinear but *convex* f and g_i on ground set $X = \mathbb{R}^n$:

$$\min \{ f(\mathbf{x}) : \mathbf{g}(\mathbf{x}) \leq \mathbf{o}\,, \mathbf{x} \in \mathbb{R}^n \} \,.$$

Under *Slater's condition* of *strict feasibility*

there exists an $\hat{\mathbf{x}} \in \mathbb{R}^n$ such that $g_i(\hat{\mathbf{x}}) < 0\,,$ all i

have again zero duality gap and *strong duality:*

there exists a dual-optimal solution $\mathbf{y}^* \in \mathbb{R}^m_+$ such that

$$\Theta(\mathbf{y}^*) = \max_{\mathbf{y} \in \mathbb{R}^m_+} \Theta(\mathbf{y}) = \min_{\mathbf{x} \in M} f(\mathbf{x})\,.$$

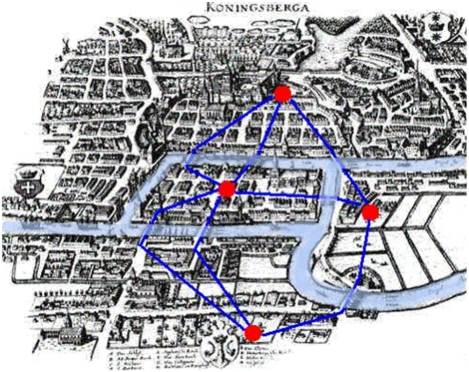
Lagrange-duality bound is tight;
can be used for optimality certificate.

GRAPHS: CONNECTIVITY & ALGORITHMS

Courtesy Dr. Michael Kahr

Seven Bridges of Königsberg (Kaliningrad)

- problem lead to the foundations of graph theory by Leonhard Euler in 1736
- find a walk through the city crossing each of the seven bridges exactly once!



Graph Representation in Algorithms

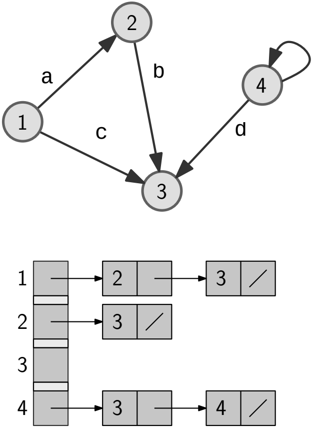


Figure 1: Digraph with adjacency list

- **adjacency matrix:**
  $|V| \times |V|$ matrix

	1	2	3	4
1	0	1	1	0
2	0	0	1	0
3	0	0	0	0
4	0	0	1	1

- **adjacency list**:
  - less memory requirements
  - but no fast way to determine if arc (u, v) exists

Predecessor Subgraph

Representation for paths / trees / forests in a directed graph, i.e., when we consider at most one incoming arc for each node:

- predecessor π_v for node v
- $R \subset V$: set of root nodes

Predecessor Subgraph $G_\pi = (V_\pi, A_\pi)$:

- $V_\pi = \{v \in V : \pi_v \text{ is defined}\} \cup R$
- $A_\pi = \{(\pi_v, v) \in A : v \in V_\pi \setminus R\}$

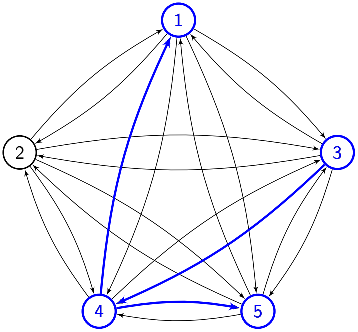
Example: predecessor subgraph G_π with $R = \{3\}$ and

v	1	2	3	4	5
π_v	4	/	/	3	4

- $V_\pi = \{1, 3, 4, 5\}$
- $A_\pi = \{(3, 4), (4, 1), (4, 5)\}$

Graph G:

- $V = \{1, 2, 3, 4, 5\}$
- $A = \{(i, j) : i, j \in V, i \neq j\}$

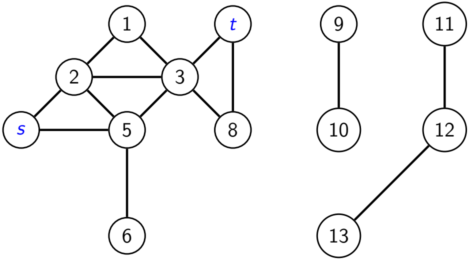


Connectivity

Definition 15 ($s - t$ connectivity problem)

Given a graph and two nodes s and t. Is there a path between s and t?

- Also known as the maze-solving problem (nodes = rooms, edges = hallways). Start in room s and find your way to room t.

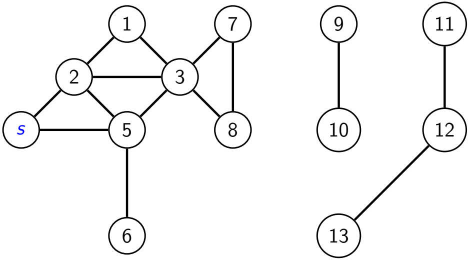


- For small graphs we can answer it easily by visual inspection. But ...

Are the two black nodes connected?

Connected Components

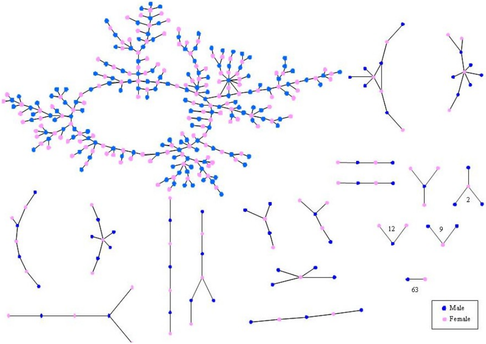
- We could solve the $s-t$ connectivity problem by finding the *connected components* of a graph which gives us all nodes reachable from s.



- Connected component containing node s: $\{1, 2, 3, s, 5, 6, 7, 8\}$

Example: Social Networks – Giant Components

- Large social networks are not necessarily connected, but they usually contain a giant component, e.g.:



- Nodes: 537 students of an American high school
- Edges: romantic relationships within an 18-month period[1]

[1] Bearman et al. Chains of Affection: The Structure of Adolescent Romantic and Sexual Networks, American Journal of Sociology 110 (1), pp.44-91, 2004

Example: Social Networks – Giant Components

Consider the friendship network:

- You are in the same component as your friends, their parents, friends of the parents, etc.
- Hence, in your component there are people who do not even live geographically close to you, do not share the same language; some probably never left their "village".
- How many such giant components exist in the friendship network?

Small-World Phenomenon

Experiment performed by Stanley Milgram in the 1960s:

- *Hypothesis:* people are connected in global friendship networks by short chains of friends
- *Setting:*
  - 296 randomly chosen "starters"
  - "target" person (a stockbroker from Boston)
  - starters were asked to send a letter to someone they know personally on the first-name basis and who might know the target person
  - this is recursively repeated until target is reached
- *Question:*
  - What is the average (or median) distance from starter to target?
  - It was about 6! But only 64 chains reached the target.

Six Degrees of Separation

Any two people on Earth are six or fewer acquaintance links apart.

Examples of Small-World Phenomenon

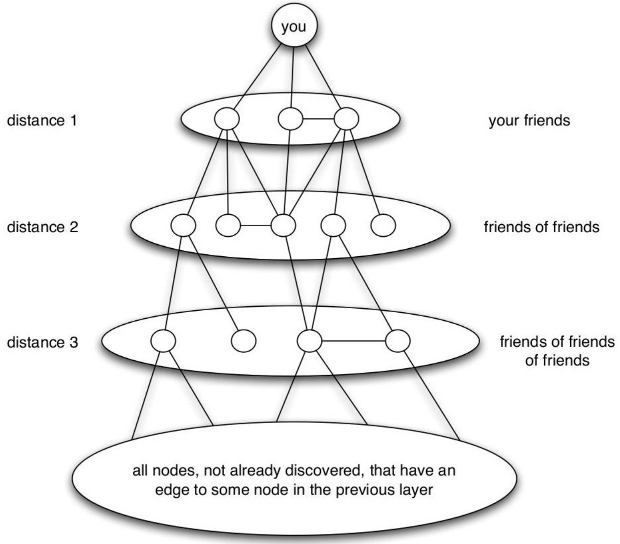
- **Erdös Number**: Erdös wrote around 1,500 mathematical articles in his lifetime, mostly co-written. He had 511 direct collaborators; these are the people with Erdös number 1; their coauthors have Erdös number 2; etc.

- Fewest number of "hops" in a communication network

Definition 16 (Distance)

The *distance* between two nodes s and t in $G = (V, E)$ is defined as the smallest number of edges in a path between s and t. If s and t are not connected, the distance is equal to ∞.

Calculate Distance



Breadth-First Search (BFS)

Idea of the algorithm: Explore all possible directions outwards from root node s, "layer by layer":

1. Start with node s in layer 0.
2. Add all nodes adjacent to s to layer 1.
3. Add all nodes to layer 2 that are adjacent to a node in layer 1 and are not yet visited, etc.
4. Continue until no new nodes are found, or until we find t.

Question: Are s and t connected?
Answer: true, if t is found, false otherwise.

BFS Algorithm

- layer $L_0 = \{s\}$
- layer L_1 = all neighbors of L_0
- layer L_2 = all nodes that do not belong to L_0 or L_1, and that are adjacent to a node in L_1
- ...
- layer L_{i+1} = all nodes that do not belong to an earlier layer, and that are adjacent to a node in L_i

Theorem 17

For each i, L_i consists of all nodes with distance exactly i from S. There is a path from s to t if and only if t appears in some layer.

Shortest Paths

Shortest Path

We are given

- a directed graph $G = (V, A)$,
- a function $w : A \to \mathbb{R}$ that assigns a weight/length/cost to each arc

Definition 1 (Shortest Path Problem (SPP))

The *shortest path problem* is the problem of finding a path with *minimal total weight* between two given nodes in a graph.

Remarks:

- fundamental problem of combinatorial optimization
- often arises as subproblem to other problems

Weight of (Shortest) Path

Definition 2 (Weight of (shortest) path)

The *weight* of a path $P = (v_1, a_1, v_2, a_2, \ldots, a_{k-1}, v_k)$ is given by

$$w(P) = \sum_{i=1}^{k-1} w(a_i) = \sum_{i=1}^{k-1} w_{v_i v_{i+1}}$$

The *shortest-path weight* from node u to v is defined as

$$\delta_{uv} = \begin{cases} \min\{w(P) : P = (u, \ldots, v)\} & \text{if there is a path from } u \text{ to } v \\ \infty & \text{otherwise.} \end{cases}$$

Typical Problem Variants

- no weights (or $w_{ij} = 1$ for all $(i, j) \in A$) $\Rightarrow$ BFS
- weighted directed graphs $\rightarrow$ topic of this section
- if no negative weights exist, same algorithms apply to undirected graphs: replace each edge by two arcs, set $w_{ij} = w_{ji}$

- **single-source SPP**: single source OR target v is specified.
  Find a shortest path $v \rightsquigarrow u$ (or $u \rightsquigarrow v$) for all $u \in V$!
- **single-pair SPP**: source u AND target v are specified.
  Find a shortest path $u \rightsquigarrow v$!
- **all-pairs SPP**:
  Find a shortest path $u \rightsquigarrow v$ for ALL pairs $u, v \in V$!

Note: no algorithms known for single-pair SPP that have better runtime complexity than best algorithms for single-source SPP.

Complexity

Remark: if cycles with total negative weight exist:

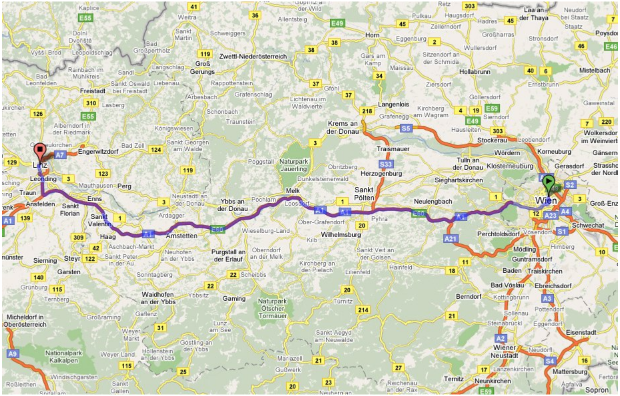
- shortest walk (possibly repeating nodes/arcs) cannot be defined
- shortest path (without repeating nodes/arcs) still exists

Theorem 3

- *If cycles with total negative weight exist, so far there is* ***no polynomial time*** *algorithm for the SPP.*
- *There are polynomial time algorithms which are able to detect a negative weight cycle, but in such a case do not give a solution to the SPP.*
- *If* ***no*** *cycles with total negative weight exist, the SPP can be solved in polynomial time.*

Applications

- Google Maps, car navigation systems



- routing in telecommunication networks
- facility layout design
- VLSI (Very-Large-Scale Integration) chip design

Bellman's Equations

Suppose we wish to find the length of the shortest paths from some specified source node s to all other nodes. These shortest path lengths must satisfy the following optimality conditions:

Definition 4 (Bellman's Equations)

We are given a graph G with arc weights w, source node $s \in V$, and shortest path lengths $\delta_u := \delta_{su}$ from source s to all other nodes $u \in V$. If there are no negative weight cycles in the network, then

$$\delta_s = 0,$$
$$\delta_j = \min_{k \in V \setminus \{j\}} \{\delta_k + w_{kj}\}, \forall j \in V \setminus \{s\}.$$

Tree of Shortest Paths

Theorem 5

If the path $P = (v_1, a_1, v_2, a_2, \ldots, a_{k-1}, v_k)$ *is a shortest path from* v_1 *to* v_k*, then for every* $q = 2, 3, \ldots, k-1$*, the subpath* $(v_1, a_1, v_2, a_2, \ldots, a_{q-1}, v_q)$ *is a shortest path from* v_1 *to* v_q*.*

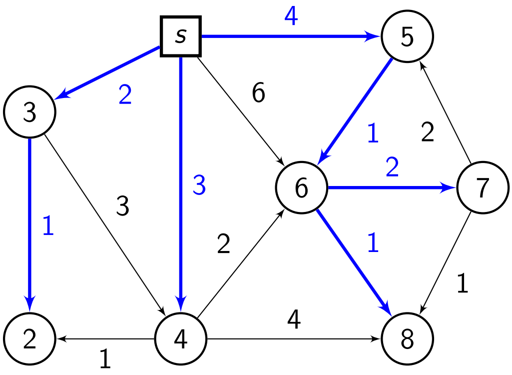
Corollary 6

Let the vector $\boldsymbol{\delta}$ *represent the shortest path distances. Then, a directed path* P *from node* s *to node* k *is a shortest path if and only if* $\delta_j = \delta_i + w_{ij}$ *for each arc* $(i, j) \in P$*.*

Corollary 7

If the network contains no negative weight cycles, then there exists a tree with root s *and including all nodes* V*, such that the path in the tree from* s *to each other node is a shortest path (shortest path tree).*

Example: Tree of Shortest Paths



- shortest path arcs from a tree rooted in source s
- $\delta_s = 0,\ \delta_2 = 3,\ \delta_3 = 2,\ \delta_4 = 3,\ \delta_5 = 4,\ \delta_6 = 5,\ \delta_7 = 7,\ \delta_8 = 6$

Overview of Algorithms

- **single-source SPP**:
  - Dijkstra's Algorithm: non-negative weights, simple implementation with runtime $\mathcal{O}(|V|^2)$
  - Bellman-Ford algorithm: arbitrary weights, detects negative weight cycles, runtime $\mathcal{O}(|V| \cdot |A|)$
- **all-pairs SPP**:
  - Floyd-Warshall Algorithm: runtime $\mathcal{O}(|V|^3)$
  - Johnson's Algorithm: runtime $\mathcal{O}(|V|^2 \log |V| + |V| \cdot |A|)$

Dijkstra's Algorithm

only for graphs with non-negative arc weights!

Idea:

- the algorithm starts from the source node and computes (upper) bounds on the shortest-path weight for all adjacent nodes
- repeat
  1. choose unfinished node k with minimal shortest-path weight δ_k and fix it permanently
  2. check if there is a shorter path to nodes adjacent to k when coming directly from k

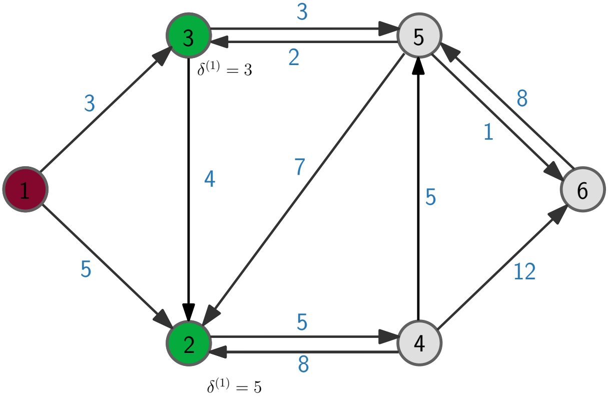
Dijkstra's Algorithm

Input: Digraph $G = (V, A)$ with $w_{ij} \geq 0, \forall (i, j) \in A$, source node s
Output: Shortest path tree with weights $\boldsymbol{\delta}$ and predecessors $\boldsymbol{\pi}$

$\delta_s := 0$
$\delta_v := \infty$ for all $v \in V \setminus \{s\}$
$F := \emptyset$
while $F \neq V$ **do** `// as long as there are unfinished nodes`
  $u := \arg\min_{v \in V \setminus F} \ \delta_v$ `// selects node with minimal` δ
  $F := F \cup \{u\}$
  forall $(u, v) \in A$, $v \notin F$ **do** `// check arcs adjacent to` u
    if $\delta_u + w_{uv} < \delta_v$ **then** `// shorter path found!`
      $\delta_v := \delta_u + w_{uv}$
      $\pi_v := u$

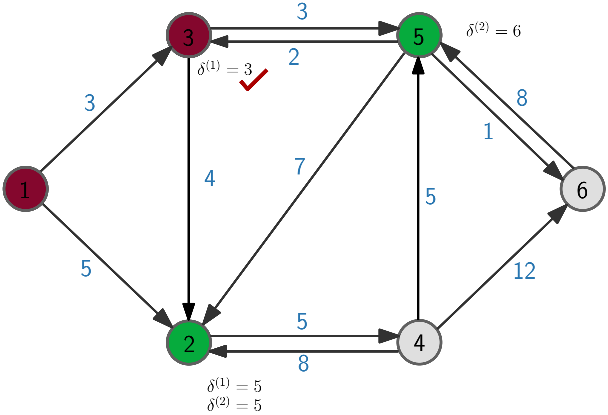
Example: Dijkstra's Algorithm

Source node = 1, finished nodes, updated nodes



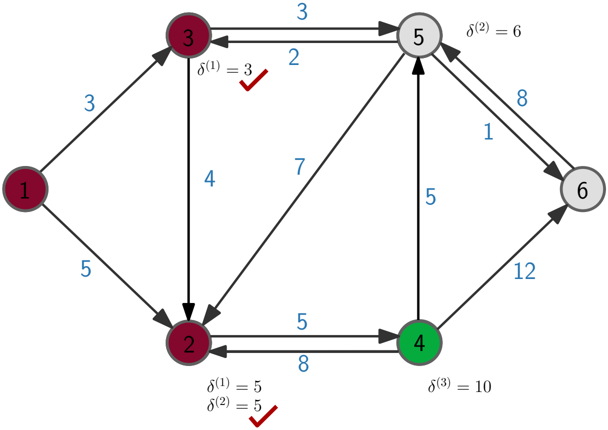
Example: Dijkstra's Algorithm

Source node = 1, finished nodes, updated nodes



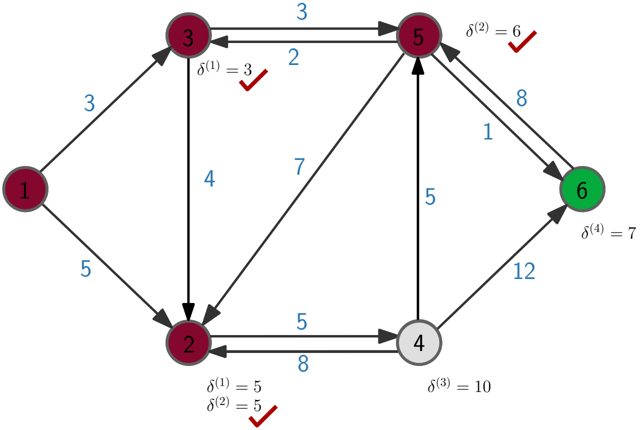
Example: Dijkstra's Algorithm

Source node = 1, finished nodes, updated nodes



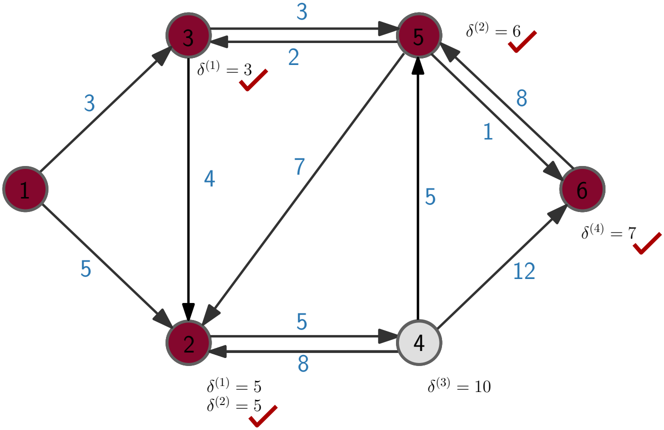
Example: Dijkstra's Algorithm

Source node = 1, finished nodes, updated nodes



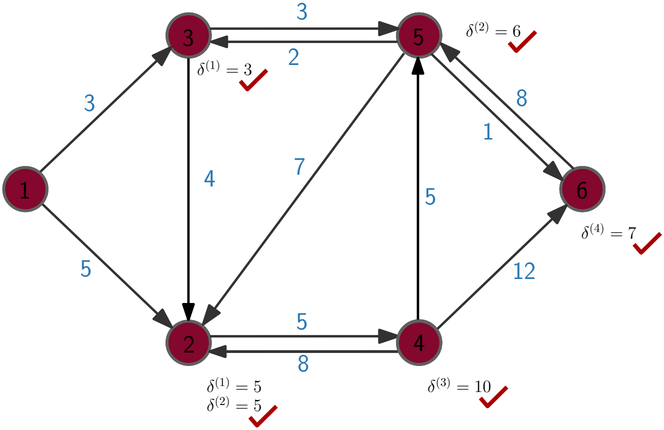
Example: Dijkstra's Algorithm

Source node $= 1$, finished nodes, updated nodes



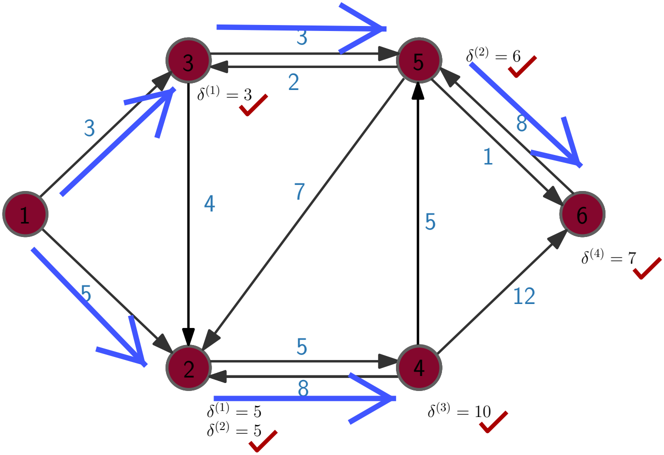
Example: Dijkstra's Algorithm

Source node = 1, finished nodes, updated nodes



Example: Dijkstra's Algorithm

Source node = 1, finished nodes, updated nodes



Shortest (s,t)-path — flow formulation

View graph as a network with source s and terminal (sink) t.

Send a flow of size $F > 0$ through network.

Decide on amount/share $f_{ij} \geq 0$ of F on all arcs (i,j) such that no flow lost — Kirchhoff's law: at all vertices v, inflow = outflow

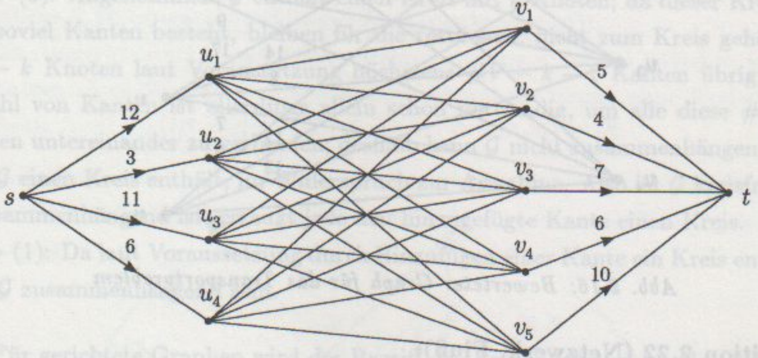
$$\sum_{i:(i,v)\in A} f_{iv} = \sum_{j:(v,j)\in A} f_{vj} \quad \text{for all } v \in V \setminus \{s,t\}\,.$$

Send F from source s:

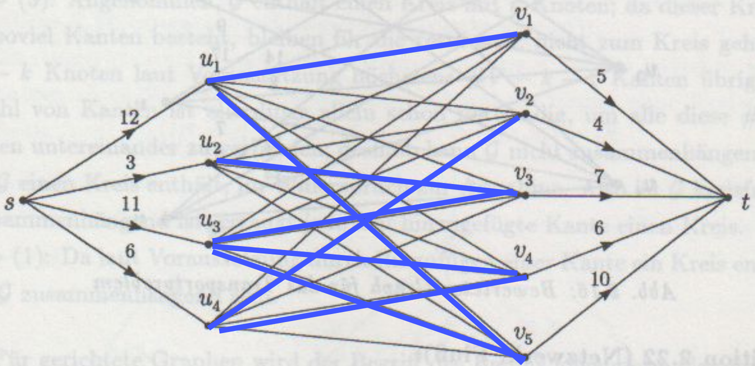
$$\sum_{j:(s,j)\in A} f_{sj} = F\,.$$

Receive F in terminal t:

$$\sum_{i:(i,t)\in A} f_{iv} = F\,.$$



s
t
u_1
u_2
u_3
u_4
v_1
v_2
v_3
v_4
v_5
12
3
11
6
5
4
7
6
10



s
t
u_1
u_2
u_3
u_4
v_1
v_2
v_3
v_4
v_5
12
3
11
6
5
4
7
6
10

Incidence matrix and flows

Recall incidence matrix $\mathsf{M} = [m_{va}]$ is $V \times A$ such that

$$m_{va} = \begin{cases} +1 & \text{if } a \text{ starts in } v \\ -1 & \text{if } a \text{ ends in } v \end{cases}$$

with s first and t last row of M.

Then flow $[f_a]_{a \in A} \in \mathbb{R}_+^{|A|}$ is a vector such that (!)

$$\mathsf{M} * \mathbf{f} = \begin{bmatrix} F \\ 0 \\ \vdots \\ 0 \\ -F \end{bmatrix}.$$

Particular case $F = 1$ enough for SPP formulation.

Enter edge weights ...

Shortest path now achieved by flow solution $\mathbf{f}^*$ minimizing cost

$$\min \quad \mathbf{w}^\top \mathbf{f} = \sum_{a \in A} w_a f_a = \sum_{(i,j) \in A} w_{ij} f_{ij} \,.$$

So arrive at an LP

$$\begin{array}{rl} \min & \mathbf{w}^\top \mathbf{f} \\ \text{s.t.} & \mathsf{M} * \mathbf{f} = \mathbf{b} := [1, 0, \ldots, 0, -1]^\top \\ & \mathbf{f} \geq \mathbf{o} \end{array}$$

Last (t) row of system $\mathsf{M} * \mathbf{f} = \mathbf{b}$ is redundant (!).

Dual LP of reduced LP reads (!)

$$\begin{array}{ll} \max & y_s \\ \text{s.t.} & y_i - y_j \leq w_{ij} \quad \text{for all } (i,j) \in A \,. \end{array}$$

Can be solved in $\leq |V|$ iterations of primal/dual simplex algorithm if $\mathbf{w} \geq \mathbf{o}$. Shortest path determined by arcs $a = (i,j)$ with $f_a^* > 0$.

Variable flows and edge capacities

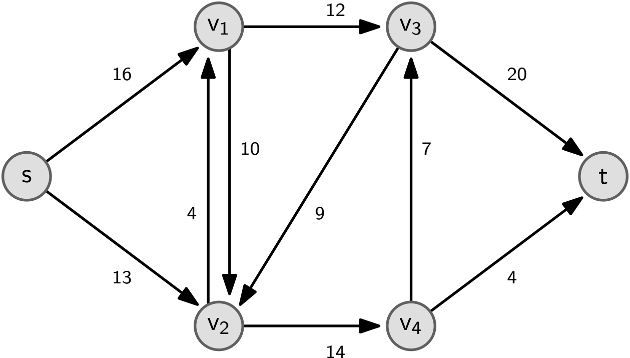
Let flow F now be a variable, write v instead. Otherwise, same formulation via incidence matrix:

$$\mathsf{M}\mathbf{f} = \begin{bmatrix} v \\ 0 \\ \vdots \\ 0 \\ -v \end{bmatrix}$$

or collecting all variables on the left,

$$\mathsf{M}\mathbf{f} + v\mathbf{d} = \mathbf{o} \quad \text{with } \mathbf{d} = [-1, 0, \ldots, 0, 1]^\top$$

plus capacity constraints on edges: $0 \leq f_a \leq k_a$ for all $a \in A$.



s
v_1
v_2
v_3
v_4
t
16
13
12
10
4
9
14
7
20
4

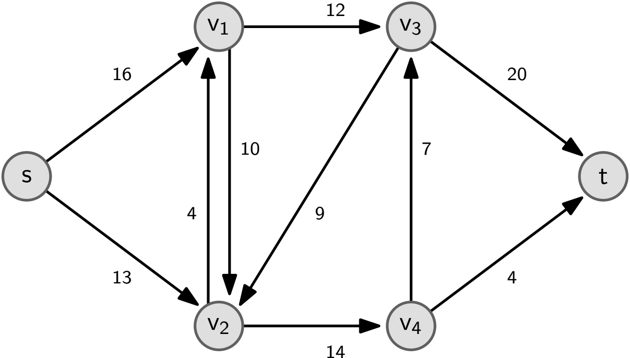
Write it as large inequality system

$$\left[\begin{array}{c|c} \mathsf{M} & \mathbf{d} \\ -\mathsf{M} & -\mathbf{d} \\ \mathsf{I} & \mathbf{o} \\ -\mathsf{I} & \mathbf{o} \end{array}\right] \begin{bmatrix} \mathbf{f} \\ v \end{bmatrix} \le \begin{bmatrix} \mathbf{o} \\ \mathbf{o} \\ \mathbf{k} \\ \mathbf{o} \end{bmatrix}$$

of inequalities in the form $\mathsf{A}^\top \mathbf{y} \le \mathbf{c}$ with $\mathbf{y}^\top = [\mathbf{f}^\top \,|\, v]$.

First two blocks are $\mathsf{M}\mathbf{f} + v\mathbf{d} \le \mathbf{o}$ and $\mathsf{M}\mathbf{f} + v\mathbf{d} \ge \mathbf{o}$, together $\mathsf{M}\mathbf{f} + v\mathbf{d} = \mathbf{o}$.

Third block is $\mathbf{f} \le \mathbf{k}$, fourth is $\mathbf{f} \ge \mathbf{o}$, together $0 \le f_a \le k_a$.



s
v_1
v_2
v_3
v_4
t
16
13
12
10
4
9
14
7
20
4

Maximal flow as a (dual) LP

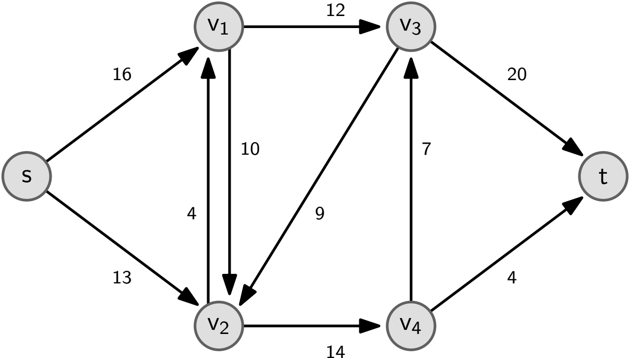
What is the maximal flow through this capacitated network ?

$$v = \left[\mathbf{o}^\top \,|\, \mathbf{1}\right] \begin{bmatrix} \mathbf{f} \\ v \end{bmatrix} = \mathbf{b}^\top \mathbf{y} \;\rightarrow\; \text{max !}$$
$$\text{subject to } \mathsf{A}^\top \mathbf{y} \;\leq\; \mathbf{c}$$

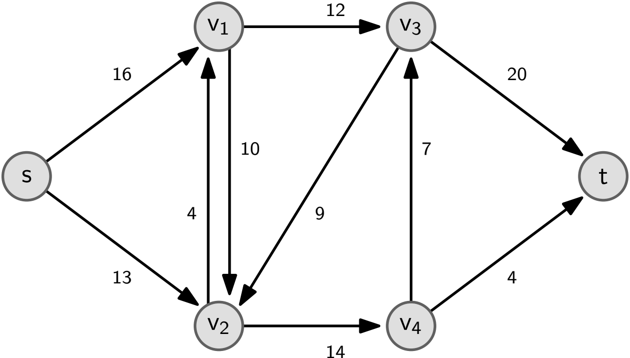
recognized as dual of another LP, namely

$$\begin{aligned} \mathbf{c}^\top \mathbf{x} &\;\rightarrow\; \text{min !} \\ \text{subject to } \mathsf{A}\mathbf{x} &\;=\; \mathbf{b} \\ \mathbf{x} &\;\geq\; \mathbf{o}\,. \end{aligned}$$

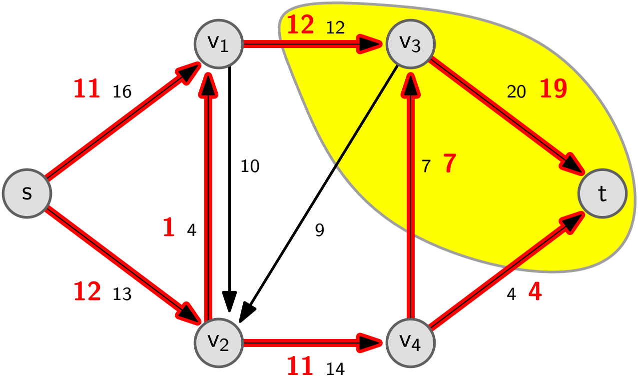
Interpretation ?



s
v_1
v_2
v_3
v_4
t
16
13
12
10
4
9
14
7
20
4



s
v_1
v_2
v_3
v_4
t
16
13
12
10
4
9
14
7
20
4


v1
v3
s
t
v2
v4
11 16
12 12
20 19
10
7 7
1 4
9
12 13
4 4
11 14

What is the related primal LP ?

$$\mathsf{A}^\top = \begin{bmatrix} \mathsf{M} & | & \mathbf{d} \\ -\mathsf{M} & | & -\mathbf{d} \\ \mathsf{I} & | & \mathbf{o} \\ -\mathsf{I} & | & \mathbf{o} \end{bmatrix} \quad \Rightarrow \quad \mathsf{A} = \begin{bmatrix} \mathsf{M}^\top & | & -\mathsf{M}^\top & | & \mathsf{I} & | & -\mathsf{I} \\ \mathbf{d}^\top & | & -\mathbf{d}^\top & | & \mathbf{o}^\top & | & \mathbf{o}^\top \end{bmatrix}.$$

And

$$\mathbf{c} = \begin{bmatrix} \mathbf{o} \\ \mathbf{o} \\ \mathbf{k} \\ \mathbf{o} \end{bmatrix}, \text{ (primal) variables } \mathbf{x} = \begin{bmatrix} \mathbf{u}^+ \\ \mathbf{u}^- \\ \mathbf{z} \\ \mathbf{s} \end{bmatrix} \quad \Rightarrow \quad \mathbf{c}^\top \mathbf{x} = \mathbf{k}^\top \mathbf{z}\,.$$

Constraints $\mathsf{A}\mathbf{x} = \mathbf{b}$ give with $\mathbf{u} = \mathbf{u}^+ - \mathbf{u}^-$ now $\mathsf{M}^\top \mathbf{u} + \mathbf{z} - \mathbf{s} = \mathbf{o}$

$$\text{or} \qquad u_i - u_j + z_{ij} \geq 0\,, \text{ all } (i,j) \in A\,,$$

while $\mathbf{d}^\top \mathbf{u} = -u_s + u_t = 1$ and $z_{ij} \geq 0$, all $(i,j) \in A$.

Primal LP describes a network cut

$$\begin{aligned} \sum_{(i,j)\in A} k_{ij} z_{ij} &\to \text{ min !} \\ \text{subject to } u_i - u_j + z_{ij} &\geq 0\,, \\ -u_s + u_t &= 1\,, \mathbf{u} \in \mathbb{R}^{|V|}, \mathbf{z} \in \mathbb{R}_+^{|E|}\,. \end{aligned}$$

Suppose have a partition $V = W \cup (V \setminus W)$ of vertices with $s \in W$, $t \notin W$ – a *cut* in the network.

Then define $u_i = 0$ if $i \in W$ and $u_i = 1$ otherwise, and

$$z_{ij} = \begin{cases} 1\,, & \text{if } i \in W, j \notin W \\ 0\,, & \text{otherwise.} \end{cases}$$

Then $(\mathbf{u}, \mathbf{z})$ are feasible for above LP and *cut capacity* is

$$k(W) = \mathbf{k}^\top \mathbf{z} = \sum_{(i,j)\in A, i\in W, j\notin W} k_{ij}\,.$$

Max-flow/min-cut theorem

Weak LP duality implies:

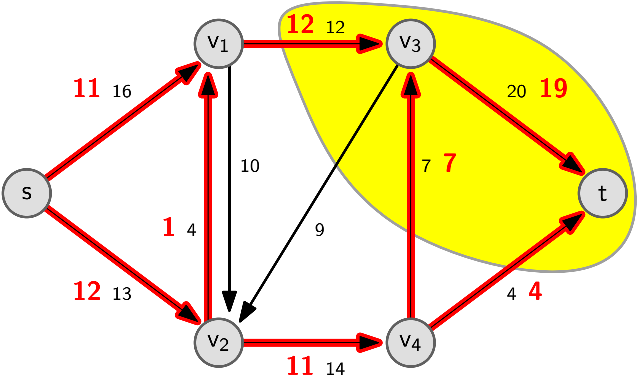
every flow $v = \mathbf{b}^\top \mathbf{y} \leq \mathbf{c}^\top \mathbf{x} = k(W)$ for any cut W.

So also the maximal flow cannot exceed minimal cut capacity. Strong LP duality (!) actually establishes equality:

The maximum flow size v^* in a network equals the capacity $k(W^*)$ of a minimal cut W^*.

Optimality conditions for the maximizing flow $\mathbf{f}^*$ read

$$f_a^* = \begin{cases} 0 & \text{for all } a = (i,j) \text{ with } i \notin W^*, j \in W^* \\ k_a & \text{for all } a = (i,j) \text{ with } i \in W^*, j \notin W^* . \end{cases}$$



s
v_1
v_2
v_3
v_4
t
11 16
12 13
12 12
1 4
10
9
20 19
7 7
11 14
4 4

MAXIMUM FLOWS

Courtesy Dr. Michael Kahr

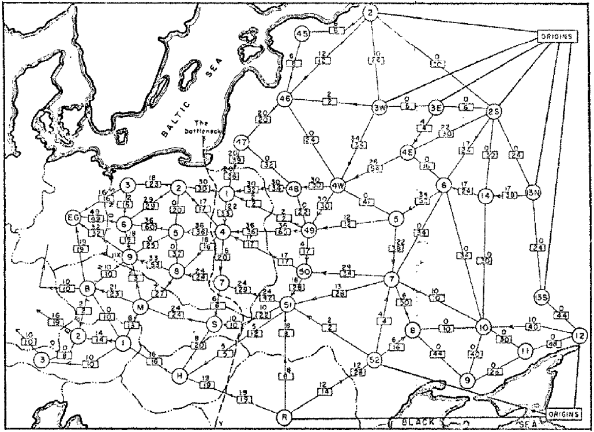
History of the Maximum Flow Problem

A secret report by Harris and Ross (1955)

- "Fundamentals of a Method for Evaluating Rail Net Capacities", written for the US Air Force, Pentagon downgraded it to "unclassified" in 1999
- solves a maximum flow problem in the railway network in Western Soviet Union / Eastern Europe
- "Consider a rail network connecting two cities by way of a number of intermediate cities, where each link of the network has a number assigned to it representing its capacity. Assuming a steady state condition, find a maximal flow from one given city to the other."
- the interest of Harris and Ross was not to find a maximum flow, but rather a minimum cut ("interdiction") of the Soviet railway system.

- Ted Harris (1966): "We were studying rail transportation in consultation with a retired army general, Frank Ross, ... We thought of modeling a rail system as a network. At first it didn't make sense, because there's no reason why the crossing point of two lines should be a special sort of node. But Ross realized that, in the region we were studying, the divisions (little administrative districts) should be the nodes. The link between two adjacent nodes represents the total transportation capacity between them. This made a reasonable and manageable model for our rail system."
- For the data they refer to secret C.I.A. reports. After aggregation of railway divisions to nodes, the network had 44 nodes and 105 (undirected) edges.
- A heuristic algorithm is proposed and applied to the railway network: It yields a flow of value 163,000 tons from sources in the Soviet Union to destinations in Eastern European satellite countries, together with a cut with capacity 163,000 tons.

Maximum Flow / Minimum Cut



Harris, Ross: *Fundamentals of a Method for Evaluating Rail Net Capacities*, Research Memorandum RM-1537, 1955