7. Dual flows and algorithms

- Duality review
- Minimum-cost flow dual
- Specialized flow duals
- Max-flow problems
- LP solvers
- Wrap-up

Laurent Lessard (www.laurentlessard.com)

Duality review

Every LP has a dual, which is also an LP.

- Every primal constraint corresponds to a dual variable
- Every primal variable corresponds to a dual constraint

Minimization	Maximization
Nonnegative variable $\geq$	Inequality constraint $\leq$
Nonpositive variable $\leq$	Inequality constraint $\geq$
Free variable	Equality constraint $=$
Inequality constraint $\geq$	Nonnegative variable $\geq$
Inequality constraint $\leq$	Nonpositive variable $\leq$
Equality constraint $=$	Free Variable

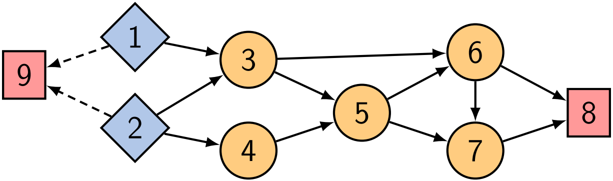
Duality review

$$
\begin{array}{llll}
\max\limits_{x} & c^{\mathsf{T}}x & & \text{(maximization)} \\
\text{s.t.} & Ax \le b & & \text{(constraint } \le\text{)} \\
& x \ge 0 & & \text{(variable } \ge\text{)}
\end{array}
\qquad
\begin{array}{llll}
\min\limits_{\lambda} & b^{\mathsf{T}}\lambda & & \text{(minimization)} \\
\text{s.t.} & \lambda \ge 0 & & \text{(variable } \ge\text{)} \\
& A^{\mathsf{T}}\lambda \ge c & & \text{(constraint } \ge\text{)}
\end{array}
$$

LP with every possible variable and constraint:

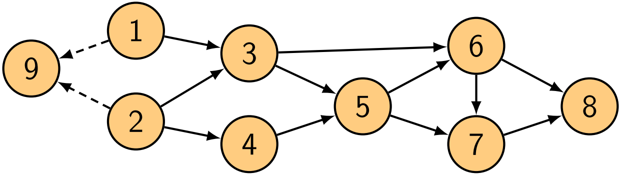
$$
\begin{array}{ll}
\max\limits_{x,y,z} & c^{\mathsf{T}}x + d^{\mathsf{T}}y + f^{\mathsf{T}}z \\
\text{s.t.} & Ax + By + Cz \le p \\
& Dx + Ey + Fz \ge q \\
& Gx + Hy + Jz = r \\
& x \ge 0 \\
& y \le 0 \\
& z \text{ free}
\end{array}
\qquad
\begin{array}{ll}
\min\limits_{\lambda,\eta,\mu} & p^{\mathsf{T}}\lambda + q^{\mathsf{T}}\eta + r^{\mathsf{T}}\mu \\
& \lambda \ge 0 \\
& \eta \le 0 \\
& \mu \text{ free} \\
& A^{\mathsf{T}}\lambda + D^{\mathsf{T}}\eta + G^{\mathsf{T}}\mu \ge c \\
& B^{\mathsf{T}}\lambda + E^{\mathsf{T}}\eta + H^{\mathsf{T}}\mu \le d \\
& C^{\mathsf{T}}\lambda + F^{\mathsf{T}}\eta + J^{\mathsf{T}}\mu = f
\end{array}
$$

Minimum-cost flow problems



- **Flow capacity constraints**: $0 \le x_{ij} \le q_{ij} \qquad \forall (i,j) \in \mathcal{E}$.
- **Supply constraint**: $\sum_{j \in \mathcal{N}} x_{ij} = s_i \qquad \forall i \in \mathcal{A}$.
- **Demand constraint**: $\sum_{i \in \mathcal{N}} x_{ij} = d_j \qquad \forall j \in \mathcal{B}$.
- **Flow conservation**: $\sum_{i \in \mathcal{N}} x_{ik} = \sum_{j \in \mathcal{N}} x_{kj} \qquad \forall k \in \mathcal{R}$.
- **Total cost**: $\sum_{(i,j) \in \mathcal{E}} c_{ij} x_{ij}$.

Minimum-cost flow problems



- **Capacity constraints**: $0 \le x_{ij} \le q_{ij} \qquad \forall (i,j) \in \mathcal{E}.$
- **Balance constraint**: $\sum_{j \in \mathcal{N}} x_{ij} = b_i \qquad \forall i \in \mathcal{N}.$
- **Minimize total cost**: $\sum_{(i,j) \in \mathcal{E}} c_{ij} x_{ij}$

Alternate formulation: every node is treated the same way. Either $b_i > 0$ (source), $b_i < 0$ (sink), or $b_i = 0$ (relay). Also, assume $\sum_i b_i = 0$.

Dual of minimum-cost flow problems

$$\begin{array}{lll} \min\limits_{x} & c^{\mathsf{T}} x & \color{red}{\text{(minimization)}} \\ \text{s.t.} & Ax = b & \color{red}{\text{(constraint } =)} \\ & x \le q & \color{red}{\text{(constraint } \le)} \\ & x \ge 0 & \color{red}{\text{(variable } \ge)} \end{array} \qquad \begin{array}{lll} \max\limits_{\mu,\eta} & b^{\mathsf{T}}\mu + q^{\mathsf{T}}\eta & \color{red}{\text{(maximization)}} \\ \text{s.t.} & \mu \text{ free} & \color{red}{\text{(variable free)}} \\ & \eta \le 0 & \color{red}{\text{(variable } \le)} \\ & A^{\mathsf{T}}\mu + \eta \le c & \color{red}{\text{(constraint } \le)} \end{array}$$

- balance constraints (at nodes)
- capacity constraints (on edges)
- flow variables (on edges)

- dual variables (at nodes)
- dual variables (each edge)
- dual constraints (each edge)

Next: what does the dual mean for: transportation? planning? max-flow?

Transportation

$$\begin{array}{lll} \min\limits_{x} & c^\mathsf{T} x & \text{(minimization)} \\ \text{s.t.} & Ax = b & \text{(constraint } =) \\ & \cancel{x \le q} & \text{(constraint } \le) \\ & x \ge 0 & \text{(variable } \ge) \end{array}$$

$$\begin{array}{lll} \max\limits_{\mu,\eta} & b^\mathsf{T}\mu + \cancel{q^\mathsf{T}\eta} & \text{(maximization)} \\ \text{s.t.} & \mu \text{ free} & \text{(variable free)} \\ & \cancel{\eta \le 0} & \text{(variable } \le) \\ & A^\mathsf{T}\mu + \cancel{\eta} \le c & \text{(constraint } \le) \end{array}$$

- balance constraints (at nodes)
- ~~capacity constraints (on edges)~~
- flow variables (on edges)

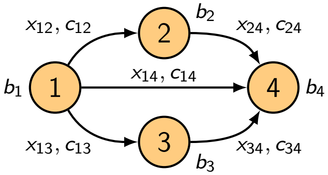
- dual variables (at nodes)
- ~~dual variables (each edge)~~
- dual constraints (each edge)

Transportation/transshipment/assignment problems: no capacity constraints on edges

Transportation (primal)

$$
\begin{array}{llll}
\min\limits_{x} & c^\mathsf{T} x & \text{(minimization)} \\
\text{s.t.} & Ax = b & \text{(constraint =)} \\
& x \ge 0 & \text{(variable} \ge\text{)}
\end{array}
\qquad
\begin{array}{llll}
\max\limits_{\mu} & b^\mathsf{T} \mu & \text{(maximization)} \\
\text{s.t.} & \mu \text{ free} & \text{(variable free)} \\
& A^\mathsf{T}\mu \le c & \text{(constraint} \le\text{)}
\end{array}
$$

$$
\begin{aligned}
\min_{x} \quad & c_{12}x_{12} + c_{13}x_{13} + c_{14}x_{14} + c_{23}x_{23} + c_{34}x_{34} \\
\text{s.t.} \quad & x_{12} + x_{13} + x_{14} = b_1 \\
& -x_{12} + x_{24} = b_2 \\
& -x_{13} + x_{34} = b_3 \\
& -x_{14} - x_{24} - x_{34} = b_4 \\
& x_{ij} \ge 0 \quad \forall i, j
\end{aligned}
$$

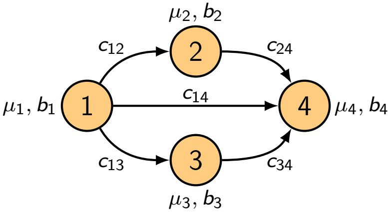


- x_{ij} are flow amounts along edges.
- Node constraints: flow is conserved and supply/demand is met.
- Edges have transportation cost. Pick x_{ij} to minimize total cost.

Transportation (dual)

$$\begin{aligned} \min_x \quad & c^\mathsf{T} x & \color{red}{\text{(minimization)}} \\ \text{s.t.} \quad & Ax = b & \color{red}{\text{(constraint } =)} \\ & x \ge 0 & \color{red}{\text{(variable } \ge)} \end{aligned} \qquad \begin{aligned} \max_\mu \quad & b^\mathsf{T} \mu & \color{red}{\text{(maximization)}} \\ \text{s.t.} \quad & \mu \text{ free} & \color{red}{\text{(variable free)}} \\ & A^\mathsf{T} \mu \le c & \color{red}{\text{(constraint } \le)} \end{aligned}$$

$$\begin{aligned} \max_\mu \quad & b_1\mu_1 + b_2\mu_2 + b_3\mu_3 + b_4\mu_4 \\ \text{s.t.} \quad & \mu_1 - \mu_2 \le c_{12} \\ & \mu_1 - \mu_3 \le c_{13} \\ & \mu_1 - \mu_4 \le c_{14} \\ & \mu_2 - \mu_4 \le c_{24} \\ & \mu_3 - \mu_4 \le c_{34} \end{aligned}$$



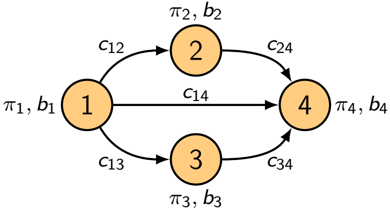
A shipping company wants to get in on this business.

- will buy commodity from sources (to alleviate supply)
- will sell commodity to destinations (to satisfy demand)
- at node i, the buy/sell price will be $\pi_i = -\mu_i$.

Transportation (dual)

$$\begin{array}{lll} \min\limits_{x} & c^{\mathsf{T}} x & \color{red}{\text{(minimization)}} \\ \text{s.t.} & Ax = b & \color{red}{\text{(constraint } =)} \\ & x \geq 0 & \color{red}{\text{(variable } \geq)} \end{array} \qquad \begin{array}{lll} \max\limits_{\pi} & -b^{\mathsf{T}} \pi & \color{red}{\text{(maximization)}} \\ \text{s.t.} & \pi \text{ free} & \color{red}{\text{(variable free)}} \\ & -A^{\mathsf{T}} \pi \leq c & \color{red}{\text{(constraint } \leq)} \end{array}$$

$$\begin{array}{ll} \max\limits_{\pi} & -(b_1\pi_1 + b_2\pi_2 + b_3\pi_3 + b_4\pi_4) \\ \text{s.t.} & \pi_2 - \pi_1 \leq c_{12} \\ & \pi_3 - \pi_1 \leq c_{13} \\ & \pi_4 - \pi_1 \leq c_{14} \\ & \pi_4 - \pi_2 \leq c_{24} \\ & \pi_4 - \pi_3 \leq c_{34} \end{array}$$



- π_i is buy/sell price of commodity at node i (shift-invariant!).
- Edge constraints: ensures the prices are competitive. e.g. if we had $\pi_2 - \pi_1 > c_{12}$, it would be cheaper to transport it ourselves!
- Pick prices π_i to maximize total profit.

Transportation summary

Primal problem:

- Pick how much commodity flows along each edge of the network to minimize the total transportation cost while satisfying supply/demand constraints.
- If each supply/demand b_i is integral, flows will be integral.

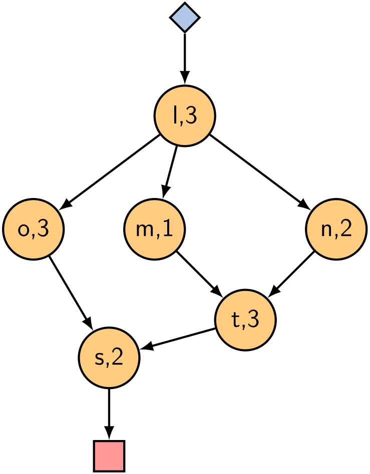
Dual problem:

- Pick the buy/sell price for the commodity at each node of the network to maximize the total profit while ensuring that the prices are competitive.
- If each edge cost c_{ij} is integral, prices will be integral.

Longest path

Recall the house-building example, a longest-path problem.

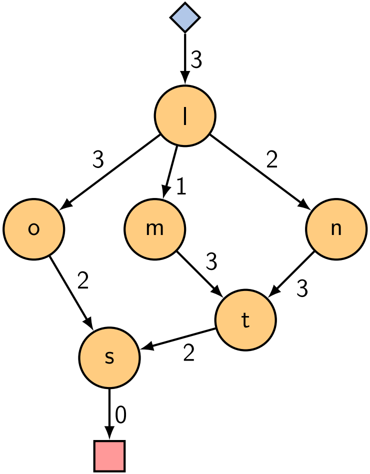
- Add source and sink nodes



Longest path

Recall the house-building example, a longest-path problem.

- Add source and sink nodes
- Move times out of nodes and onto preceding edges
- Solve longest-path problem

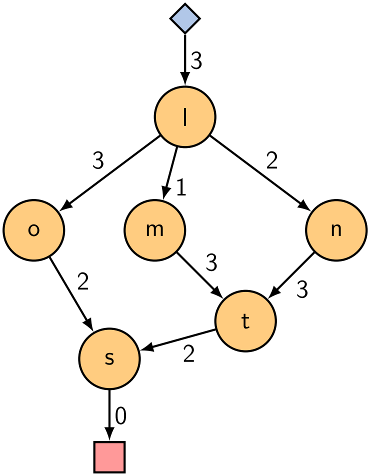


Longest path (primal)

Recall the house-building example, a longest-path problem.

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & c^{\mathsf{T}} x \\ \text{subject to:} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

unit flow: $b = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ -1 \end{bmatrix}$

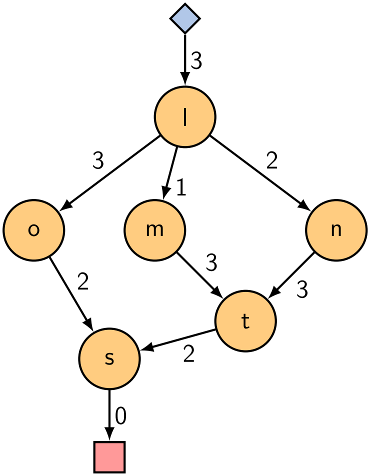


Longest path (dual)

Recall the house-building example, a longest-path problem.

$$\begin{aligned} \underset{\mu}{\text{minimize}} \quad & b^\mathsf{T}\mu \\ \text{subject to:} \quad & A^\mathsf{T}\mu \geq c \end{aligned}$$

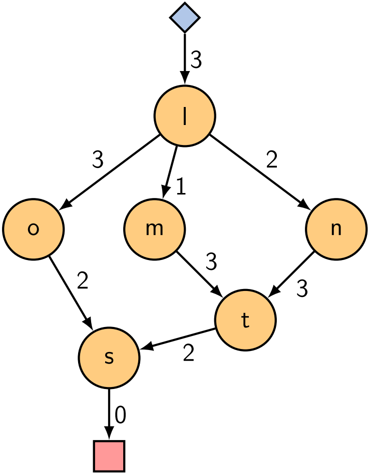
- using same trick as before, define: $t_i = -\mu_i$



Longest path (dual)

Recall the house-building example, a longest-path problem.

$$
\begin{aligned}
\underset{t}{\text{minimize}} \quad & t_{\text{end}} - t_{\text{start}} \\
\text{subject to:} \quad & t_l - t_{\text{start}} \geq 3 \\
& t_o - t_l \geq 3 \\
& t_m - t_l \geq 1 \\
& \dots
\end{aligned}
$$



Precisely the alternative problem formulation we deduced in class!

Longest path (dual)

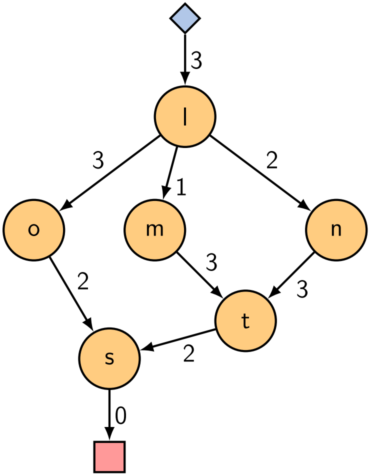
Recall the house-building example, a longest-path problem.

Key players:

- Primal variables: $x_{ij} \in \{0, 1\}$
- Dual constraints: $t_j - t_i \geq c_{ij}$

Complementary slackness:

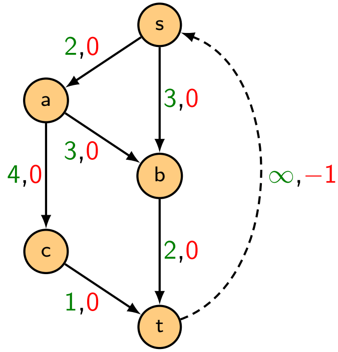
- if $x_{ij} = 1$ then $t_j - t_i = c_{ij}$ (longest path corresponds to tight time constraints)
- if $t_j - t_i > c_{ij}$ then $x_{ij} = 0$ (this path has slack)



Max-flow

We are given a directed graph and edge capacities. Find the maximum flow that we can push from source to sink.

- Edges have max capacities
- Edges have zero cost except feedback edge, with cost -1.
- Finding max flow is equivalent to finding the minimum cost flow.



(edge capacity)
(cost)

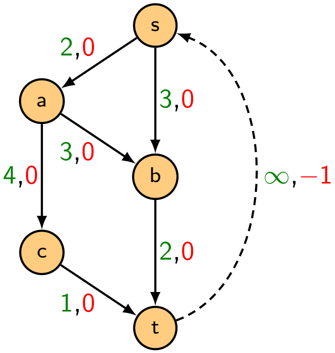
Max-flow (primal)

- **Primal problem**:

$$\max_{x_{ij}} \quad x_{ts}$$

$$\text{s.t.} \quad \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_{sa} \\ x_{sb} \\ x_{ab} \\ x_{ac} \\ x_{bt} \\ x_{ct} \\ x_{ts} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \le \begin{bmatrix} x_{sa} \\ x_{sb} \\ x_{ab} \\ x_{ac} \\ x_{bt} \\ x_{ct} \end{bmatrix} \le \begin{bmatrix} 2 \\ 3 \\ 3 \\ 4 \\ 2 \\ 1 \end{bmatrix}$$

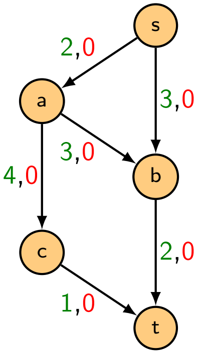


(edge capacity)
(cost)

Max-flow (dual)

- **Dual problem**:

$$
\begin{aligned}
\min_{\lambda_{ij},\mu_i} \quad & 2\lambda_{sa} + 3\lambda_{sb} + 3\lambda_{ab} + 4\lambda_{ac} + 2\lambda_{bt} + \lambda_{ct} \\
\text{s.t.} \quad & \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_s \\ \mu_a \\ \mu_b \\ \mu_c \\ \mu_t \end{bmatrix} + \begin{bmatrix} \lambda_{sa} \\ \lambda_{sb} \\ \lambda_{ab} \\ \lambda_{ac} \\ \lambda_{bt} \\ \lambda_{ct} \\ 0 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\
& \lambda_{ij} \geq 0, \quad \mu_i \text{ free}
\end{aligned}
$$



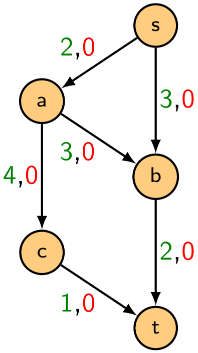
(edge capacity)
(cost)

  - μ_i are shift-invariant. Therefore we may assume $\mu_s = 0$.
  - We want each λ_{ij} small; no slack!

Max-flow (dual)

- **Dual problem**:

$$
\begin{aligned}
\min_{\lambda_{ij},\mu_i} \quad & 2\lambda_{sa} + 3\lambda_{sb} + 3\lambda_{ab} + 4\lambda_{ac} + 2\lambda_{bt} + \lambda_{ct} \\
\text{s.t.} \quad & \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_a \\ \mu_b \\ \mu_c \\ \mu_t \end{bmatrix} + \begin{bmatrix} \lambda_{sa} \\ \lambda_{sb} \\ \lambda_{ab} \\ \lambda_{ac} \\ \lambda_{bt} \\ \lambda_{ct} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\
& \lambda_{ij} \geq 0, \quad \mu_i \text{ free}
\end{aligned}
$$



(edge capacity)
(cost)

  - Rearrange constraints, isolate λ_{ij}.

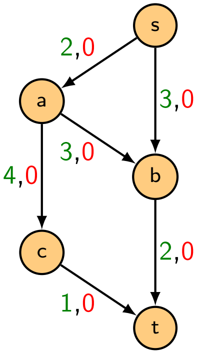
Max-flow (dual)

$\mu_i \in \{0,1\}$ for all i.

- **Dual problem**:

$$
\begin{aligned}
\min_{\lambda_{ij},\mu_i} \quad & 2\lambda_{sa} + 3\lambda_{sb} + 3\lambda_{ab} + 4\lambda_{ac} + 2\lambda_{bt} + \lambda_{ct} \\
\text{s.t.} \quad & \mu_s = 0 \\
& \mu_a - \mu_s = \lambda_{sa} \\
& \mu_b - \mu_s = \lambda_{sb} \\
& \mu_b - \mu_a = \lambda_{ab} \\
& \mu_c - \mu_a = \lambda_{ac} \\
& \mu_t - \mu_b = \lambda_{bt} \\
& \mu_t - \mu_c = \lambda_{ct} \\
& \mu_t = 1 \\
& \lambda_{ij} \ge 0, \quad \mu_i \text{ free}
\end{aligned}
$$

  - Each path, e.g. $s \to a \to c \to t$ has: $0 = \mu_s \le \mu_a \le \mu_c \le \mu_t = 1$



(edge capacity)
(cost)

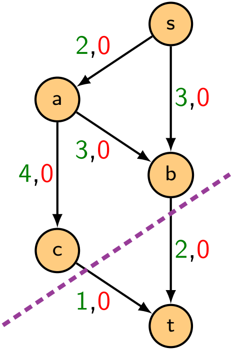
Max-flow (dual)

- **Dual problem**:

$$\begin{aligned} \min_{\lambda_{ij},\mu_i} \quad & 2\lambda_{sa} + 3\lambda_{sb} + 3\lambda_{ab} + 4\lambda_{ac} + 2\lambda_{bt} + \lambda_{ct} \\ \text{s.t.} \quad & \text{Along each path } s \to i \to \cdots \to j \to t \\ & \text{exactly one edge } p \to q \text{ is chosen.} \\ & \lambda_{pq} = 1 \text{ and } \lambda_{ij} = 0 \text{ for all other edges.} \end{aligned}$$

  - Each path is broken by *cutting* edges. We are choosing the cut with lowest total cost (**min-cut**).

$$\boxed{\text{Max flow} = \text{Min cut}}$$



(edge capacity)
(cost)

Max-flow summary

Primal problem:

- Each edge of the network has a maximum capacity.
- Pick how much commodity flows along each edge to maximize the total amount transported from the start node to the end node while obeying conservation constraints. This total amount of flow is called the **max flow**.

Dual problem:

- Find a partition of the nodes into two subsets where the first subset includes the start node and the second subset includes the end node.
- Choose the partition that minimizes the sum of capacities of all edges that connect both subsets. This total capacity is called the **min cut**.

LP solvers

Modern LP solvers are very efficient. Problems with millions of variables and/or constraints are routinely solved. Three main categories of algorithms are used in practice for solving LPs:

- **Simplex algorithms**: traverse the *surface* of the feasible polyhedron looking for the best vertex. (`Clp`, `GLPK`)
- **Interior point**: traverse the *inside* of the polyhedron and move toward the best vertex. (`GLPK`, `SCS`, `ECOS`, `Ipopt`)
- **Blended**: a custom (proprietary) mixture of simplex and interior point methods. (`CPLEX`, `Gurobi`, `Mosek`)

Simplex method

- Invented by George Dantzig in 1947.
- Named one of the "Top 10 algorithms of the 20th century" by *Computing in Science & Engineering* Magazine.
  Full list at: https://www.siam.org/pdf/news/637.pdf
- The basic idea:
  - We know the solution is a vertex of the feasible polyhedron.
  - Each vertex is characterized by the subset of the constraints that have no slack; it's just a system of linear equations!
  - Start at a vertex, then *pivot*: swap out one of the constraints in the no-slack subset so that the cost improves.
  - Do this in a systematic way that avoids cycles. When we can no longer improve, we are optimal!

Simplex method

- With m constraints in n variables, the feasible polyhedron can have roughly up to $\binom{m}{n}$ vertices, a very large number!
- A cube in n dimensions has 2^n vertices.
- By carefully designing the problem, the simplex method may visit *all the vertices*! Look up the Klee–Minty cube.
- It is not known whether there is a more clever version of simplex that is sub-exponential in the worst case.

Despite these difficulties, the simplex method works **very well** in practice. For typical problems, its performance scales linearly with m and n.

Interior point methods

- Big family of optimization algorithms, dating back to 1950–1960. Can be used for solving convex nonlinear optimization problems. Will revisit later in the course!
- Ellipsoid method when applied to LPs achieves polynomial-time convergence (Khachiyan, 1979), but typically much slower than simplex in practice.
- Specialized interior-point solvers developed for LPs in 1980–1990 are competitive with simplex method, especially for very large problems.
- Still active area of research!

Specialized algorithms

If the LP has a special form, specialized algorithms are often vastly superior to generic simplex or interior point solvers.

- **Network simplex method**: special version of simplex method for solving minimum-cost flow problems. Can be 100s of times faster than using ordinary simplex method. Polynomial worst-case, and can be called in `CPLEX`.
- **Graph searches**: Djikstra's algorithm, A* search, etc. Can be used for example to find the shortest path in a graph.
- Assignment problems: Kuhn–Munkres, auction algorithm.
- Max-flow problems: Ford–Fulkerson, Orlin's algorithm.

LP wrap-up

- Relevant courses at UW–Madison
  - CS 525: linear programming methods
  - CS 526: advanced linear programming
  - CS 577: introduction to algorithms

  These courses prove major results (e.g. zero duality gap), give detailed explanations/analyses of simplex/graph algorithms.

- External resources
  - EE 236A: Linear programming (UCLA)
    http://www.seas.ucla.edu/~vandenbe/ee236a/
  - MATH 407: Linear programming (Univ. Washington)
    https://www.math.washington.edu/~burke/crs/407/
  - 15.082J: Network optimization (MIT)
    http://ocw.mit.edu/courses/sloan-school-of-management/15-082j-network-optimization-fall-2010/