Recitation 5

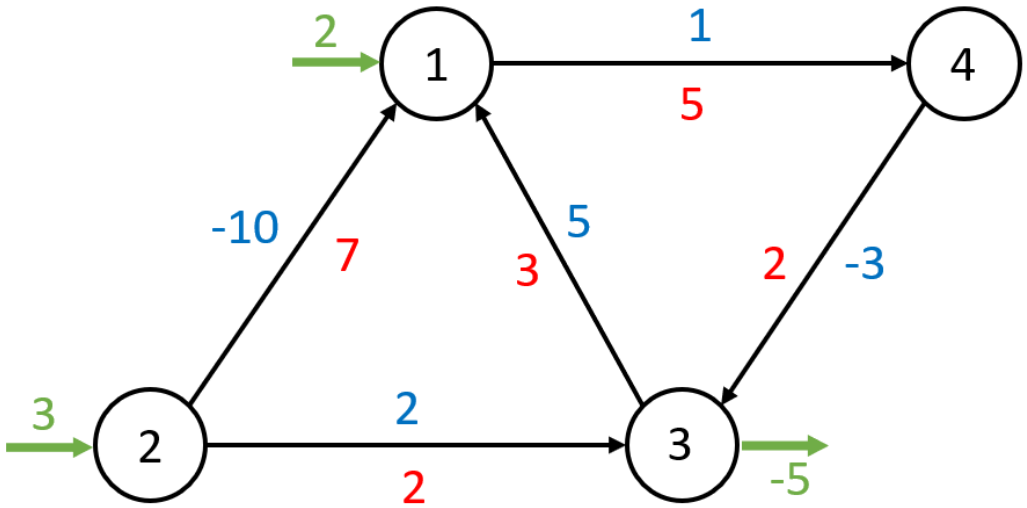
6.255 Optimization Methods

10/08/2021

Agenda

- Summary on network flows
- Formulation exercises
- (Large-scale optimization review)

Network flow



- A directed graph $\mathcal{G} = (N, A)$ (set of arcs $A \subset V \times V$)
- inflows/outflows b_i for $i \in N$ with $\sum_{i \in N} b_i = 0$.
- edge costs c_{ij} for $(i, j) \in A$
- non-negative edge capacities u_{ij} for $(i, j) \in A$ (possibly infinite)

Network flow formulation

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b_i \quad \text{for all } i \in N \\
& x_{ij} \leq u_{ij} \quad \text{for all } (i,j) \in A \\
& x_{ij} \geq 0 \quad \text{for all } (i,j) \in A
\end{aligned}
$$

Network flow formulation

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b_i \quad \text{for all } i \in N \\
& x_{ij} \leq u_{ij} \quad \text{for all } (i,j) \in A \\
& x_{ij} \geq 0 \quad \text{for all } (i,j) \in A
\end{aligned}
$$

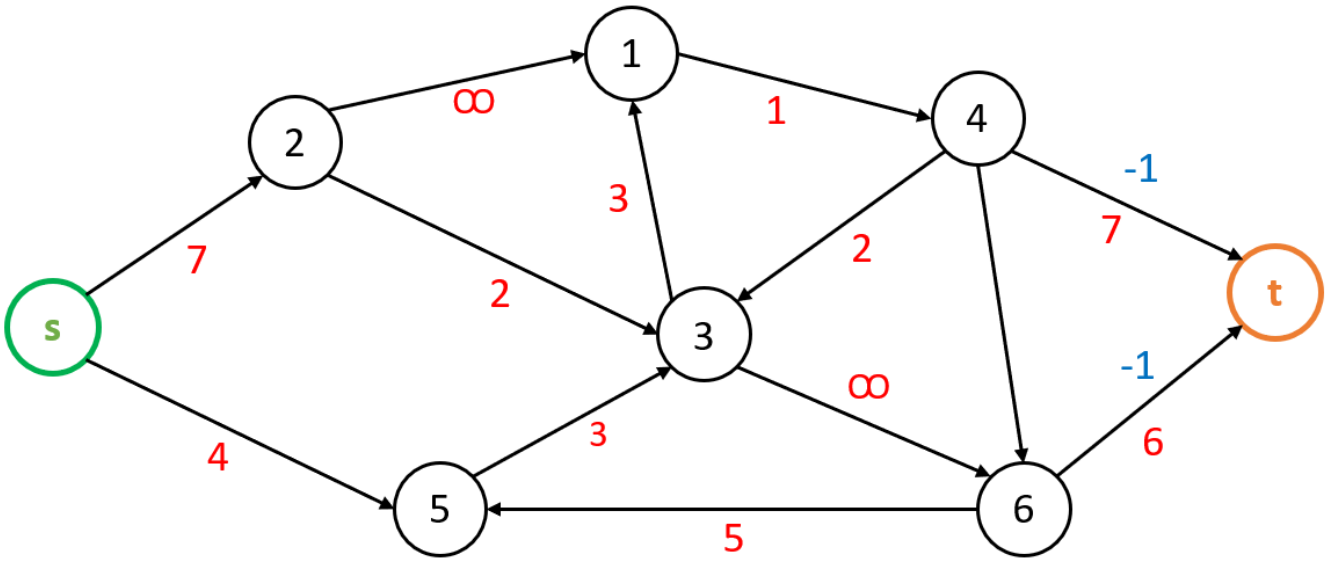
- If $b \in \mathbb{Z}$ and $u \in \mathbb{N} \cup \{+\infty\}$ then optimal flow in $\mathbb{N}$ (integer).
- Basic (feasible) solution $\leftrightarrow$ Spanning tree (feasible) solution.

Special cases of network flows

- **Circulation network**: $b = 0$ i.e. conservation of flow at any node

Special cases of network flows

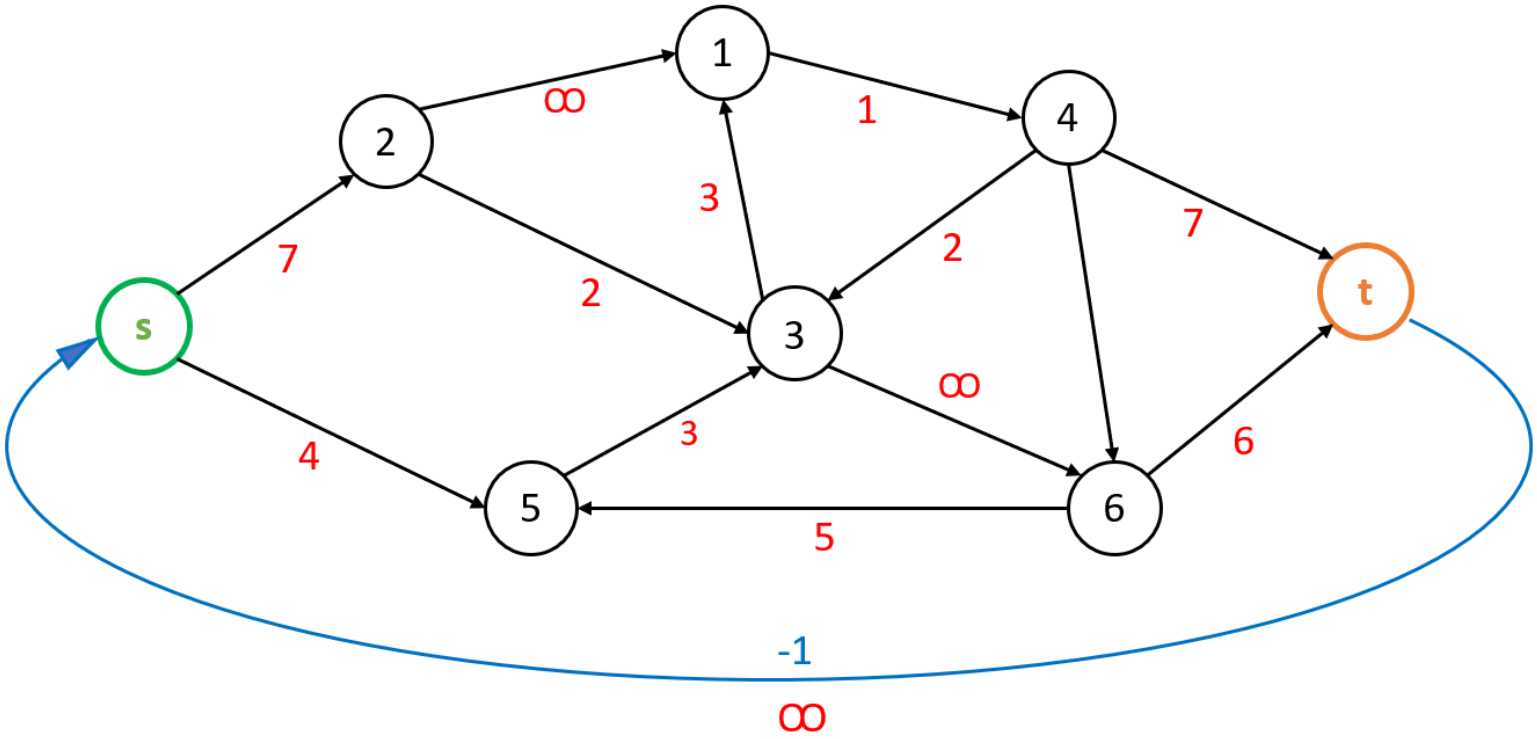
- **Circulation network**: $b = 0$ i.e. conservation of flow at any node
- **Max-flow network**:
  - $d_i(s) = 0$ and $d_o(t) = 0$
  - $b_i = 0$ for any $i \in N \setminus \{s, t\}$
  - no conservation constraint at s and t
  - objective is maximum inflow to the sink: max $\sum_{j:(j,t)\in A} x_{jt}$.



Special cases of network flows

Max-flow is a special case of circulation network problem:

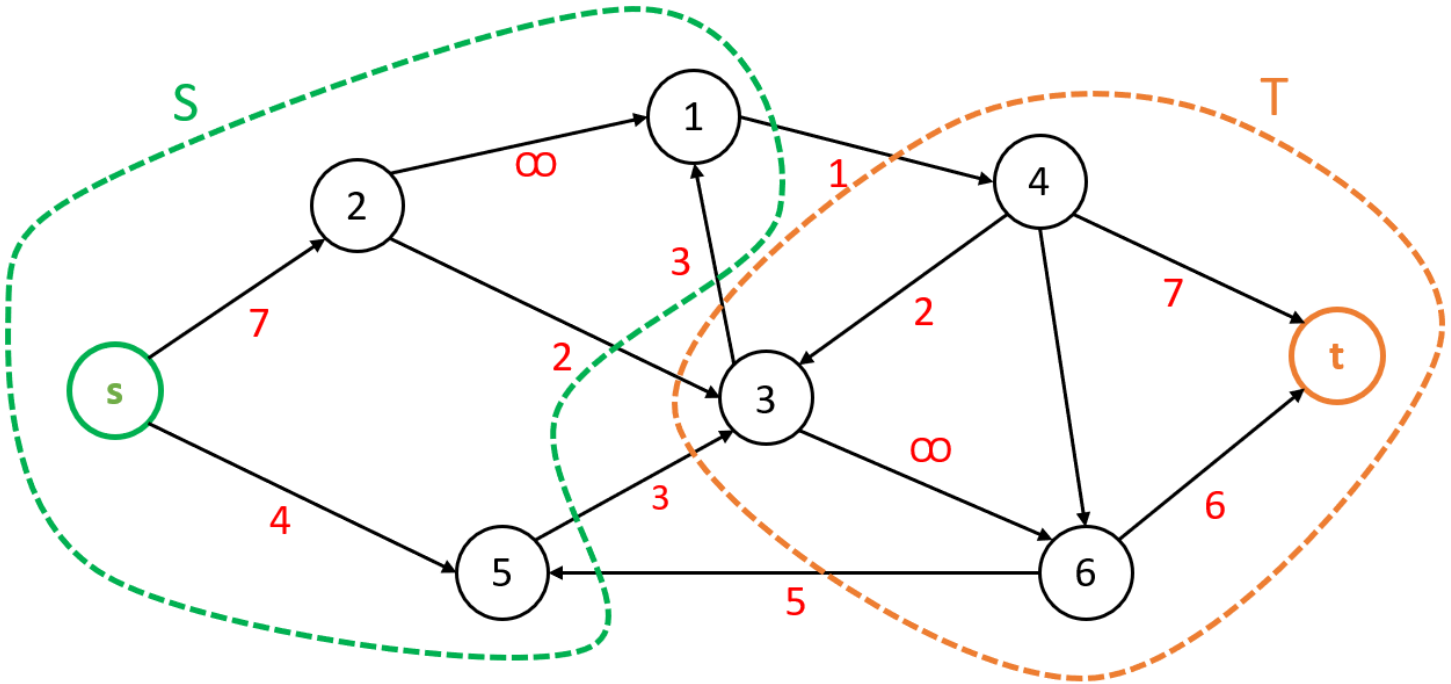
- add an uncapacitated edge from sink to source: $u_{ts} = \infty$
- zero costs for all arcs except for $c_{ts} = -1$: objective $\min -x_{ts}$.



Max-Flow = Min-Cut

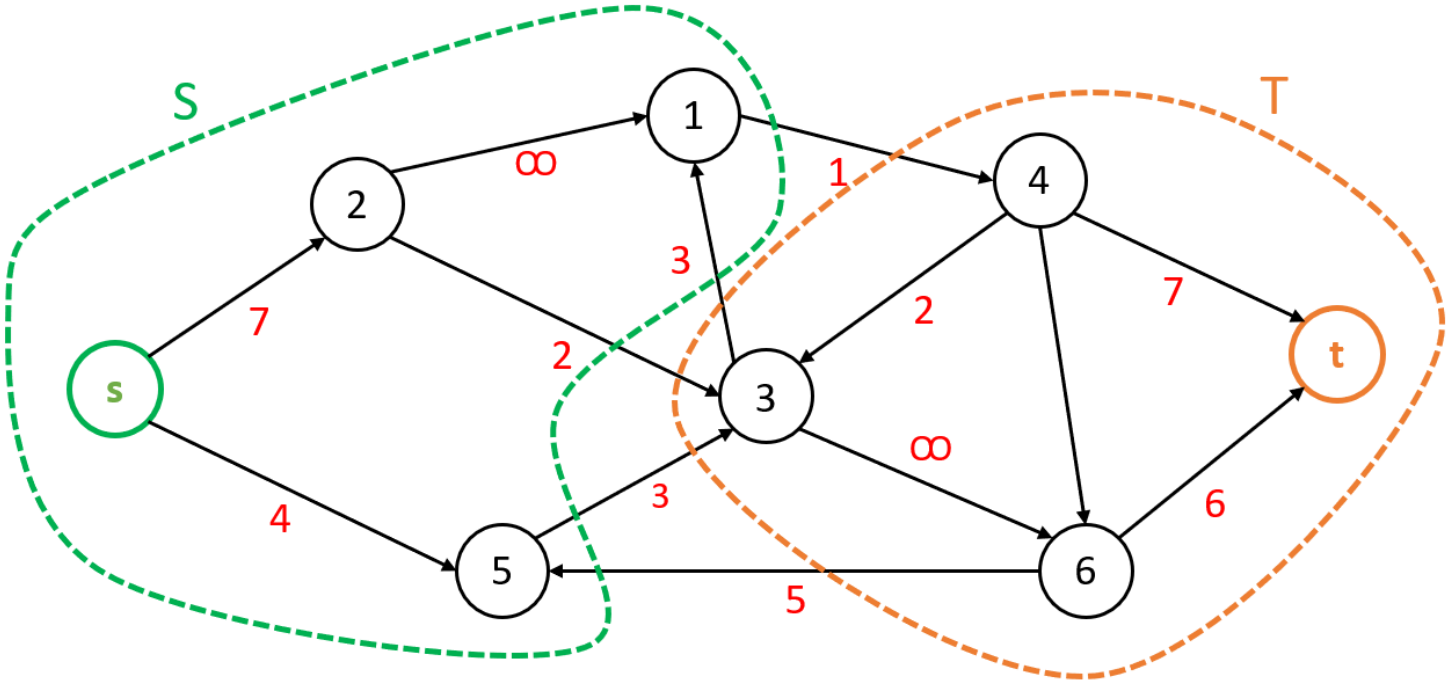
Min-Cut problem: Find $s - t$ cut with minimal value:

- a partition (S, T) of the nodes N with $s \in S$ and $t \in T$
- capacity of the cut $C(S, T) = \sum_{(i,j) \in A, i \in S, j \in T} u_{i,j}$



Max-Flow = Min-Cut

Theorem: Max $s-t$-flow = Min $s-t$-cut.



Proof (exercise): The dual of the maximum flow problem is equivalent to the min-cut problem.

Exercise 1

Consider a catering company which provides tablecloths for n consecutive days. On day i, r_i tablecloths must be provided to the customer. The catering company can buy new tablecloths for p dollars each, or launder the used ones. Laundering takes 1 day and costs f dollars. The caterer's problem is to decide how to meet the client's demand at minimum cost, starting with no tablecloths and under the assumption that any leftover tablecloths have no value. Formulate the caterer's problem as a network flow problem.

Solution

There are many valid solutions. One possibility is to have a node C_i with demand r_i , and a node D_i with supply r_i for each $i = 1, 2, 3$. Also have a node S with no supply or demand. C_i corresponds to clean tablecloths for day i, D_i corresponds to dirty tablecloths for day i, and S is a master inventory of tablecloths. Have an uncapacitated arc with cost p from S to C_i for each $i = 1, 2, 3$. This corresponds to the option of buying tablecloths. Have an uncapacitated arc with cost f from D_i to C_j for $j > i$. This corresponds to the laundering option. Finally, have an uncapacitated arc with cost 0 from D_i to S. This corresponds to discarding a dirty tablecloth.

Exercise 2:

Consider an airplane with maximum capacity C who travels from city N_1 to N_n with regional stops at cities $N_2, \ldots N_{n-1}$ in this order. A passenger who wants to go from N_i to N_j for $i < j$ can stay in the same seat during the regional flights until she/he reaches destination. Suppose we know the total passenger demand d_{ij} for traveling from N_i to N_j, and the price p_{ij} for each corresponding ticket is given for all $1 \leq i < j \leq n$. We wish to select passengers in order to maximize the total revenue for the airplane travel. Formulate this as a network flow problem.

Solution

Consider the graph $N_1 \to N_2 \to \cdots \to N_n$ with all arcs having capacity C and cost 0. We add for any $1 \leq i < j \leq n$ an arc $j \to i$ with capacity d_{ij} and cost $-p_{ij}$. Minimizing the cost on this network flow problem gives the optimal strategy: given the optimal solution (which is integer), the flow on each arc $j \to i$ will count the number of passengers traveling from N_i to N_j. More precisely, we can decompose the flow into unitary flows on cycles (circuits). Each circuit is of the form $N_i \to \cdots \to N_j \to N_i$ and corresponds to a passenger going from N_i to N_j.

Large scale optimization (review)

Consider a "fat" matrix $A \in \mathbb{R}^{m \times n}$ with large $n \gg m$. We want to solve the LP.

$$\begin{aligned} \min \quad & c^\top x \\ s.t. \quad & Ax = b \\ & x \geq 0. \end{aligned} \qquad \begin{aligned} \max \quad & b^\top p \\ s.t. \quad & A^T p \geq b \\ & p \text{ free.} \end{aligned}$$

- Too many columns.
- **Column generation**

- Too many constraints.
- **Cutting planes**

$$\begin{aligned} \min \quad & c^\top x \\ s.t. \quad & Ax = b \\ & x \geq 0. \end{aligned}$$

$$\begin{aligned} \max \quad & b^\top p \\ s.t. \quad & A^T p \geq b \\ & p \text{ free.} \end{aligned}$$

Column generation

1. Start with few columns $A_{i_1}, \ldots, A_{i_k}$ $(k \ll n)$
2. Find optimal solution x^k
3. Access to some oracle:
   - which returns some column j with $\tilde{c}_j < 0$ if exists
   - or certifies that $\tilde{c} \geq 0$.
4. If x^k not optimal in original problem, add column A_j $(i_{k+1} = j)$ and iterate from step 2 (primal simplex)

Cutting planes

1. Start with few constraints $a_{i_1}, \ldots a_{i_k}$ $(k \ll n)$
2. Find optimal solution p^k
3. Access to some oracle:
   - which returns some violated constraint j if exists $(a_j^\top p^k < b_j)$
   - or certifies that x^k feasible.
4. If x^k infeasible, add constraint j $(i_{k+1} = j)$ and iterate from step 2 (dual simplex)