DM545

Linear and Integer Programming

Lecture 11

More on Network Flows

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

1. More on Network Flows

2. Cutting Plane Algorithms

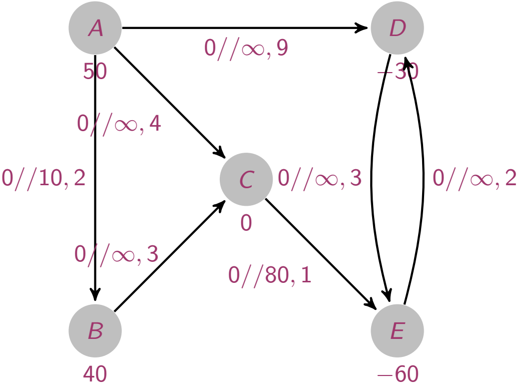
Peer Review of Assignment 1

- You will receive an anonymous report per email.
  The report will be chosen among those being present in class today.
- Comment the report without giving grades. Be picky and polite!
- Bring the report in class the next time.
- In class you will gather in pairs in a tournament-like fashion and compare the two reports
- The reports ranking last will be reviewed by the instructor and risk a no pass.

Outline

ILP in Excel

ILP can be solved in Excel!
Let's solve the min cost flow problem below:



See file `mincost.xlsx`

Is the simplex algorithm polynomial or exponential in the worst case?
Is an LP problem polynomially solvable or NP-hard?

Minimum spanning tree

Definition

Given a graph $G = (V, E)$

- a forest is a subgraph $G' = (V, E')$ containing no cycles
- a tree is a subgraph $G' = (V, E')$ that is a forest and is connected ($\exists$ a (uv)-path $\forall u, v \in V$)

Proposition

A graph $G = (V, E)$ is a tree iff

- *it is a forest containing exactly $n - 1$ edges*
- *it is an edge minimal connected graph spanning V*
- *it contains a unique path between every pair of nodes of V*
- *the addition of an edge not in E creates a unique cycle.*

Solvable via greedy algorithm (Kruskall)

$$\max \sum_{e \in E} c_e x_e \tag{1}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \qquad \text{for } 2 \leq |S| \leq n \tag{2}$$

$$x_e \geq 0 \qquad \text{for } e \in E \tag{3}$$

$$x \in \mathbb{Z}^{|E|} \tag{4}$$

Theorem

The convex hull of the incidence vectors of the forests in a graph is given by the constraints (2)-(3)

Network simplex

- Improved version of the simplex method for network flows (still not polynomial but performs well in practice)
- it goes through same basic steps at each iteration:
  finding basic variable + determining leaving variable + solving for the new basis
- executes these steps exploiting network structure without needing a simplex tableau
- Key idea: network representation of basic feasible solutions

- in min cost flow formulation one of the node constraints is redundant (summing all these constraints yields zero on both sides - $\sum_i b_i = 0$)
- with $n-1$ non redundant node constraints, we have just $n-1$ basic variables for a basic solution
  each basic variable x_{ij} represents the flow though arc ij: basic arcs
- basic arcs never form undirected cycles...
- hence they form a spanning tree

Multi-commodity flow problem

Multi-commodity flow problem ship several commodities using the same network, different origin destination pairs separate mass balance constraints, share capacity constraints, min overall flow

$$
\begin{aligned}
\min \textstyle\sum_k c^k x^k & \\
N x^k \geq b^k & \qquad \forall k \\
\textstyle\sum_k x_{ij}^k \geq u_{ij} & \qquad \forall ij \in A \\
0 \leq x_{ij}^k \leq u_{ij}^k &
\end{aligned}
$$

How does the structure of the matrix looks like? Is it still TUM?

Residual Network

Residual Network $N(x)$:
replace arc $ij \in N$ with arcs:

$$\begin{aligned} ij &: c_{ij}, r_{ij} = u_{ij} - x_{ij} \\ ji &: -c_{ij}, r_{ji} = x_{ij} \end{aligned}$$

Max Flow Algorithms

Optimality Condition

- Ford Fulkerson augmenting path algorithm $O(m|x^*|)$
- Edmonds-Karp algorithm (augment by shortest path) in $O(nm^2)$
- Dinic algorithm in layered networks $O(n^2m)$
- Karzanov's push relabel $O(n^2m)$

Min Cost Flow Algorithms

Optimality conditions: Let x be feasible flow in $N(V, A, l, u, b)$ then x is min cost flow in N iff $N(x)$ contains no directed cycle of negative cost.

- Cycle canceling algorithm with Bellman Ford Moore for negative cycles $O(nm^2UC)$
- Build up algorithms $O(n^2mM)$

Matching Algorithms

Matching: $M \subseteq E$ of pairwise non adjacent edges

- bipartite graphs
- arbitrary graphs
- cardinality (max or perfect)
- weighted

Assignment problem $\equiv$ min weighted perfect bipartite matching $\equiv$ special case of min cost flow

bipartite cardinality

Theorem

The cardinality of a max matching in a bipartite graph equals the value of a maximum (s,t)-flow in N_{st}.

$\rightsquigarrow$ Dinic $O(\sqrt{n}m)$

Theorem (Optimality condition (Berge))

A matching M in a graph G is a maximum matching iff G contains no M-augmenting path.

$\rightsquigarrow$ augmenting path $O(\min(|U|,|V|), m)$

bipartite weighted
build up algorithm $O(n^3)$
bipartite weighted: Hungarian method $O(n^3)$

minimum weight perfect matching
Edmonds $O(n^3)$

Theorem (Hall's (marriage) theorem)

A bipartite graph $B = (X, Y, E)$ *has a matching covering* X *iff:*

$$|N(U)| \geq |U| \quad \forall U \subseteq X$$

Theorem (König, Egeavary theorem)

Let $B = (X, Y, E)$ *be a bipartite graph. Let* M^* *be the maximum matching and* V^* *the minimum vertex cover:*

$$|M^*| = |V^*|$$

Outline

Valid Inequalities

- IP: $z = \max\{c^T x : x \in X\}, X = \{x : Ax \leq b, x \in \mathbb{Z}_+^n\}$
- Proposition: $\text{conv}(X) = \{x : \tilde{A}x \leq \tilde{b}, x \geq 0\}$ is a polyhedron
- LP: $z = \max\{c^T x : \tilde{A}x \leq \tilde{b}, x \geq 0\}$ would be the best formulation
- Key idea: try to approximate the best formulation.

Definition (Valid inequalities)

$ax \leq b$ is a valid inequality for $X \subseteq \mathbb{R}^n$ if $ax \leq b \; \forall x \in X$

Which are useful inequalities? and how can we find them?
How can we use them?

Example: Pre-processing

- $X = \{(x, y) : x \leq 999y; 0 \leq x \leq 5, y \in \mathbb{B}^1\}$

$$x \leq 6y$$

- $X = \{x \in \mathbb{Z}^n_+ : 13x_1 + 20x_2 + 11x_3 + 6x_4 \geq 72\}$

$$2x_1 + 2x_2 + x_3 + x_4 \geq \frac{13}{11}x_1 + \frac{20}{11}x_2 + x_3 + \frac{6}{11}x_4 \geq \frac{72}{11} \geq 6 + \frac{6}{11}$$
$$2x_1 + 2x_2 + x_3 + x_4 \geq 7$$

- UFL:

$$\sum_{i \in M} x_{ij} \leq b_j y_j \quad \forall j \in N \qquad\qquad x_{ij} \leq b_j y_j$$
$$\sum_{j \in N} x_{ij} = a_i \quad \forall i \in M \qquad\qquad x_{ij} \leq a_i$$
$$x_{ij} \geq 0, y_j \in B^n \qquad\qquad x_{ij} \leq \max\{a_i, b_j\} y_j$$

To be continued next lecture

Summary

1. More on Network Flows

2. Cutting Plane Algorithms