Applied Quantitative Logistics

Well-Solved Problems Part I: Network Flows

Simulated Annealing (SA)

Majid Sohrabi
National Research University Higher School of Economics

January 31, 2025

Properties of Easy Problems

For the moment we will be very imprecise and say that an algorithm on a graph $G = (V, E)$ with n nodes and m edges is **efficient** if, in the **worst** case, the algorithm requires $O(m^p)$ elementary calculations (such as additions, divisions, comparison, etc) for some integer p, where we assume that $m \geq n$.

Properties of Easy Problems

Consider the following combinatorial optimization problem

$$\text{COP} \qquad \min\{cx : x \in X \subseteq R^n\}$$

Definition: The Separation Problem

The Separation Problem associated with COP is the problem: Given a point $x^* \in R^n$, is $x^* \in conv(X)$? If not find an inequality $\pi x \leq \pi_0$ satisfied by all points in X, but violated by the point x^*.

Properties of Easy Problems

When examining a problem to see if it has an efficient algorithm, we will see that the following four properties often go together:

- **Efficient Optimization Property**: for a given class of problems P, there exists an efficient (polynomial) algorithm.
- **Strong Dual Property**: For the given problem class, there exists a strong dual problem $(D)\ max = \{w(u) : u \in U\}$ allowing us to obtain optimality conditions that can be quickly verified: $x^* \in X$ is optimal in P if and only if there exists $u^* \in U$ with $cx^* = w(u^*)$.
- **Efficient Separation Property**: there exists an efficient algorithm for the separation problem associated with the problem class.
- **Explicit Convex Hull Property**: A compact description of the convex hull $conv(X)$ is known, which in principle allows us to replace every instance by the linear program $\min\{cx : x \in conv(X)\}$.

Properties of Easy Problems

When considering integer programming:

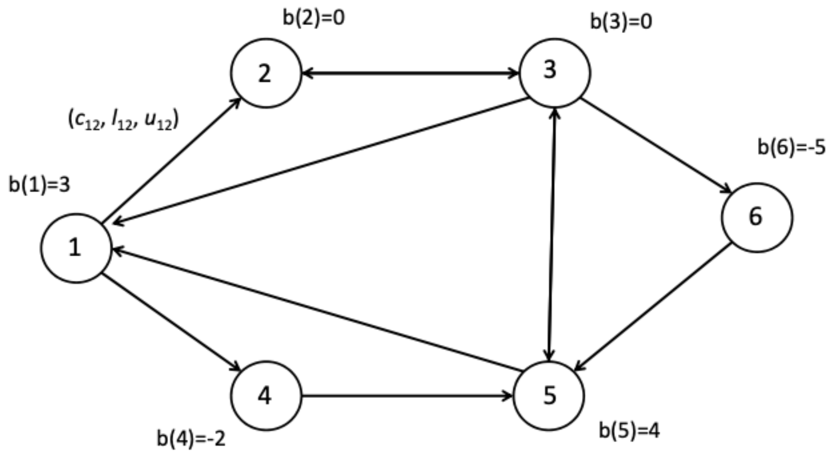
$$\text{(IP)} \qquad min\{cx: Ax \leq b, x \in Z_+^n\},$$

With integral data (A, b), it is important to know when the linear programming relaxation

$$\text{(LP)} \qquad min\{cx: Ax \leq b, x \in R_+^n\},$$

Will have an optimal solution that is integral.

Minimum Cost Flow Problem



Input:

- $G = (N, A)$: directed graph
- c_{ij}: cost per unit flow on $(i\ j) \in A$
- h_{ij} : maximum flow on $(i\ j) \in A$
- b_i: supply / demand of $i \in N$

The minimum Cost Flow Problem:

- If bi > 0, node i is a supply node, bi < 0, node i is a demand node, and bi = 0, node i is a transshipment node
- Objective: minimize the total flow cost for sending the products from the suppliers to the customers through the network
- Capacity constraints on the arcs must me respected

Minimum Cost Flow Problem

Input:

- x_{ij}: amount of flow in arc $(i\ j) \in A$

The minimum cost flow problem can be formulated as:

$$\text{minimize} \quad \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$\text{subject to} \quad \sum_{j\in V^+(i)} x_{ij} - \sum_{j\in V^-(i)} x_{ji} = b_i \qquad i \in N \qquad (1)$$

$$0 \le x_{ij} \le h_{ij} \qquad (i,j) \in A \qquad (2)$$

- Constraints (1) are known as flow conservation constraints
- A feasible solution will exist if $\sum_{i\in N} b_i = 0$

Minimum Cost Flow Problem

Proposition:

The constraint matrix A arising in a minimum cost network flow problem is totally unimodular.

Corollary:

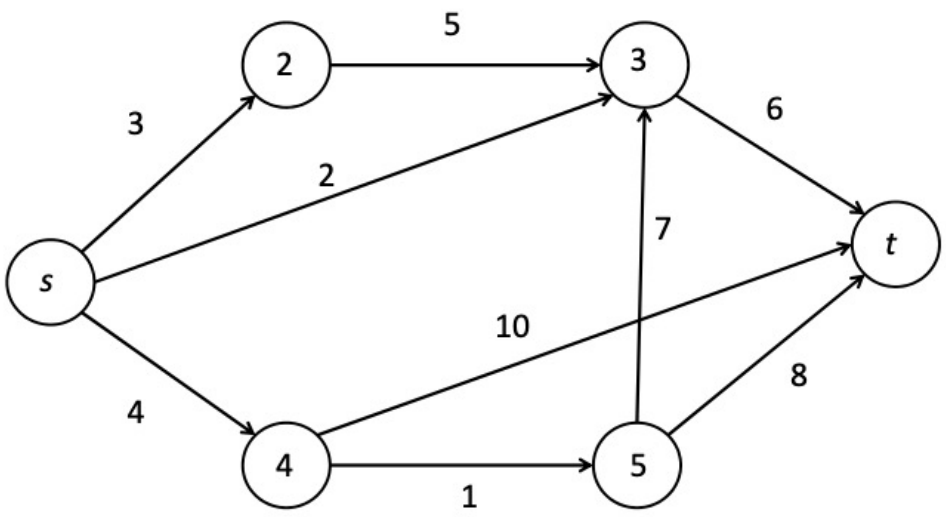
In a minimum cost network flow problem, if the demands bi and the capacities hij are integral,

- Each extreme point is integral
- The constraints describe the convex hull of integral feasible flows

Theorem: (Pioncare, 1900)

Let A be a ± 1 valued matrix, where each column has at most one $+1$ and at most -1. Then A is totally unimodular.

Shortest Path Problem



Input:

- $G = (N, A)$: directed graph
- $c_{ij} \geq 0$: length of arc $(i\ j) \in A$
- s: source node
- t: sink node

The Shortest Path Problem:

- Objective: Find the shortest path between node s and node t

Shortest Path Problem

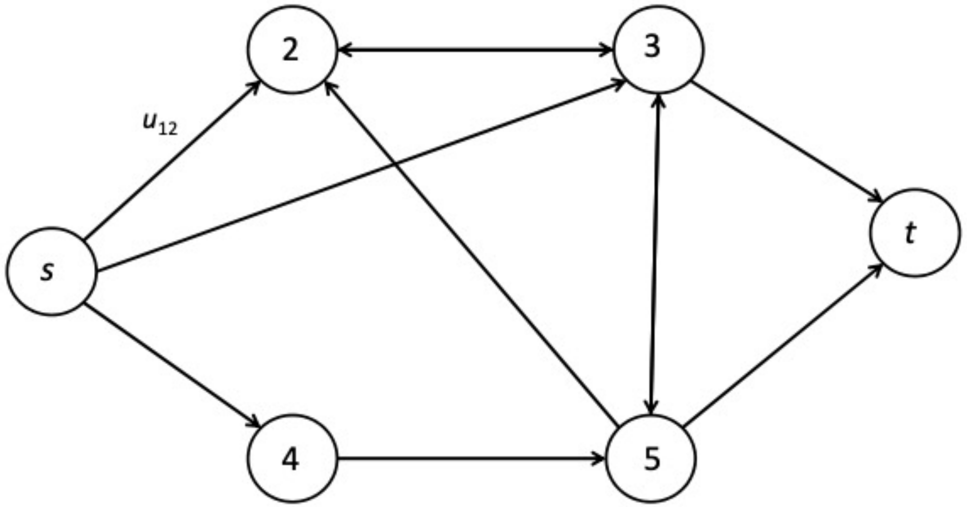
Path variables

$$x_{ij} = \begin{cases} 1 & if\ arc\ (i,j) \in A\ is\ in\ the\ shortest\ s-t\ path; \\ 0 & Otherwise \end{cases}$$

The shortest path problem can be formulated as:

$$\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in A} c_{ij} x_{ij} \\
\text{subject to} \quad & \sum_{j\in V^+(i)} x_{ij} - \sum_{j\in V^-(i)} x_{ji} = 1 \qquad i = s \\
& \sum_{j\in V^+(i)} x_{ij} - \sum_{j\in V^-(i)} x_{ji} = 0 \qquad i \in N \setminus \{s,t\} \\
& \sum_{j\in V^+(i)} x_{ij} - \sum_{j\in V^-(i)} x_{ji} = -1 \qquad i = t \\
& x_{ij} \in \{0,1\} \qquad (i,j) \in A
\end{aligned}$$

The Maximum Flow Problem



Input:

- $G = (N, A)$: directed graph
- h_{ij}: maximum flow on $(i\ j) \in A$
- s: source node
- t: sink node

The maximum $s - t$ Flow Problem:

- Objective: Find the maximum flow between node s and node t that satisfies the arc capacities

The Maximum Flow Problem

Flow variables

- x_{ij} = amount of flow in arc $(i, j) \in A$
- v = flow sent from source node s to sink node t

The maximum $s - t$ flow problem can be formulated as:

$$
\begin{array}{lll}
\text{maximize} & v & \\
\text{subject to} & \displaystyle\sum_{j \in V^+(i)} x_{ij} - \sum_{j \in V^-(i)} x_{ji} = v & i = s \\
& \displaystyle\sum_{j \in V^+(i)} x_{ij} - \sum_{j \in V^-(i)} x_{ji} = 0 & i \in N \setminus \{s, t\} \\
& \displaystyle\sum_{j \in V^+(i)} x_{ij} - \sum_{j \in V^-(i)} x_{ji} = -v & i = t \\
& 0 \leq x_{ij} \leq h_{ij} \qquad (i, j) \in A &
\end{array}
$$

The Maximum Flow Problem

Adding a backward arc from t to s, the maximum $s-t$ flow problem can be formulated as:

$$\begin{array}{lll} \text{maximize} & x_{ts} & \\ \text{subject to} & \displaystyle\sum_{j\in V^+(i)} x_{ij} - \sum_{j\in V^-(i)} x_{ji} = 0 & i \in N \\ & 0 \leq x_{ij} \leq h_{ij} & (i,j) \in A \end{array}$$

The dual is:

$$\begin{array}{lll} \text{minimze} & \displaystyle\sum_{(i,j)\in A} h_{ij} w_{ij} & \\ \text{subject to} & u_i - u_j + w_{ij} \geq 0 & (i,j) \in A \\ & u_t - u_s \geq 1 & \end{array}$$

Simulated Annealing

Refer to the notes

Thank you!

msohrabi@hse.ru

@MSohrabi_CS