Introduction to integer programming III:
Network Flow, Interval Scheduling, and Vehicle Routing Problems

Martin Branda

Charles University
Faculty of Mathematics and Physics
Department of Probability and Mathematical Statistics

Computational Aspects of Optimization

Totally unimodular matrices

Definition
A matrix A is totally unimodular (TU) iff every square submatrix of A has determinant $+1$, -1, or 0.

The linear program has an integral optimal solution for all integer r.h.s. b if and only if A is TU.

Totally unimodular matrices

... based on Laplace expansion for the determinant of a basic matrix and the Cramer rule.

Totally unimodular matrices

A set of **sufficient conditions**:

- $a_{ij} \in \{-1, 0, 1\}$ for all i, j
- Each column contains at most two nonzero coefficients, i.e. $\sum_{i=1}^{m} |a_{ij}| \le 2$,
- There exists a partitioning $M_1 \cap M_2 = \emptyset$ of the rows $1, \ldots, m$ such that each column j containing two nonzero coefficients satisfies

$$\sum_{i \in M_1} a_{ij} = \sum_{i \in M_2} a_{ij}.$$

If A is TU, then A^T and $(A|I)$ are TU.

Minimum cost network flow problem

- $G = (V, A)$ – graph with vertices V and (oriented) arcs A
- h_{ij} – arc capacity
- c_{ij} – flow cost
- b_i – demand, ASS. $\sum_i b_i = 0$
- $V^+(i) = \{k : (i,k) \in A\}$ – successors of i
- $V^-(i) = \{k : (k,i) \in A\}$ – predecessors of i

$$\begin{aligned} \min_{x_{ij}} \ & \sum_{(i,j)\in A} c_{ij}x_{ij} \\ \text{s.t.} \ & \sum_{k\in V^+(i)} x_{ik} - \sum_{k\in V^-(i)} x_{ki} = b_i, \ i \in V, \\ & 0 \le x_{ij} \le h_{ij}, \ (i,j) \in A. \end{aligned}$$

Wolsey (1998), Ex. 3.1 ($M_1 = \{1, \dots, m\}, M_2 = \emptyset$)

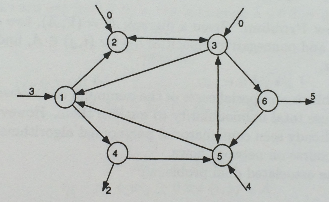


Fig. 3.1 Digraph for minimum cost network flow

equations:

x_{12}	x_{14}	x_{23}	x_{31}	x_{32}	x_{35}	x_{36}	x_{45}	x_{51}	x_{53}	x_{65}		
1	1	0	−1	0	0	0	0	−1	0	0	=	3
−1	0	1	0	−1	0	0	0	0	0	0	=	0
0	0	−1	1	1	1	1	0	0	−1	0	=	0
0	−1	0	0	0	0	0	1	0	0	0	=	−2
0	0	0	0	0	−1	0	−1	1	1	−1	=	4
0	0	0	0	0	0	−1	0	0	0	1	=	−5

Special cases

- Shortest path problem
- Critical (longest time) path problem in project scheduling (PERT = Program Evaluation and Review Technique)
- Fixed interval scheduling
- Transportation problem

Shortest path problem

Find a minimum cost $s - t$ **path** given nonnegative arc costs c_{ij}, set

- $b_i = 1$ if $i = s$,
- $b_i = -1$ if $i = t$,
- $b_i = 0$ otherwise.

Then the problem can be formulated as

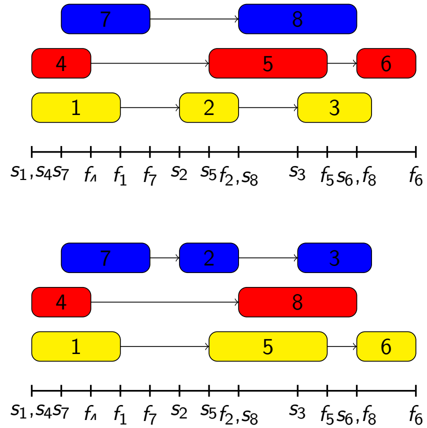
$$\begin{aligned} \min_{x_{ij}} \ & \sum_{(i,j)\in A} c_{ij}x_{ij} \\ \text{s.t.} \ & \sum_{k\in V^+(i)} x_{ik} - \sum_{k\in V^-(i)} x_{ki} = 1, \ i = s, \\ & \sum_{k\in V^+(i)} x_{ik} - \sum_{k\in V^-(i)} x_{ki} = 0, \ i \in V \setminus \{s,t\}, \\ & \sum_{k\in V^+(i)} x_{ik} - \sum_{k\in V^-(i)} x_{ki} = -1, \ i = t, \\ & 0 \le x_{ij} \le 1, \ (i,j) \in A. \end{aligned}$$

$\hat{x}_{ij} = 1$ identifies the shortest path.

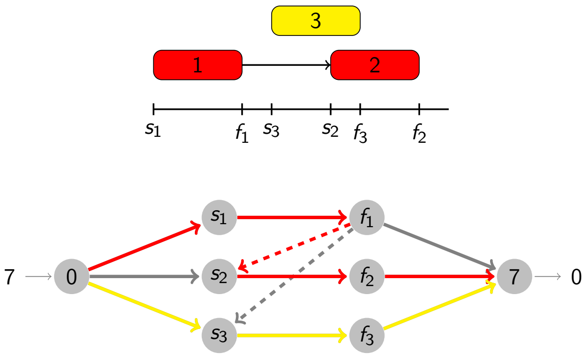
Fixed interval scheduling

Basic **Fixed interval scheduling** (FIS) problem: given J jobs with prescribed starting s_j and finishing f_j times, find a minimal number of identical machines that can process all jobs such that no processing intervals intersect.

Fixed interval scheduling



FIS – network flow reformulation



FIS – network flow reformulation

Network structure:

1. $2J+2$ **vertices** $\mathcal{V}$: $\{0, s_1, f_1, \dots s_J, f_J, 2J+1\}$; vertices 0, $2J+1$ correspond to the source and sink,
2. **oriented edges** E: $\{0, s_j\}$, $\{s_j, f_j\}$, $j \in \mathcal{J}$, $\{f_i, s_j\}$ if $f_i \le s_j$, $\{f_j, 2J+1\}$, $j \in \mathcal{J}$, $(2J+1, 0)$
3. demands: $d_0 = d_{2J+1} = 0$, $d_{s_j} = -1$, $d_{f_j} = 1$, $j \in \mathcal{J}$,
4. return edge $(2J+1, 0)$: capacity $u_{2J+1,0} = M$, $c_{2J+1,0} = 1$,
5. edge capacities $u_{uv} = 1$, and costs $c_{uv} = 0$, $(u, v) \in E \setminus (2J+1, 0)$.

Solve the min-cost network flow problem.

Traveling salesman problem

- Consider n towns and in one of them there is a traveling salesman.
- Traveling salesman must visit all towns and return back.
- For each pair of towns the traveling costs are known and the traveling salesman is looking for the cheapest route.

= Finding a Hamilton cycle in a graph with edge prices.

Assignment problem

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{1}$$

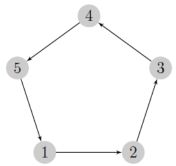
$$\sum_{i=1}^{n} x_{ij} = 1, \ j = 1, \dots, n, \tag{2}$$

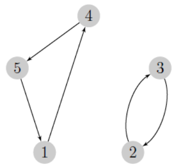
$$\sum_{j=1}^{n} x_{ij} = 1, \ i = 1, \dots, n, \tag{3}$$

$$x_{ij} \in \{0, 1\}. \tag{4}$$

We minimize the traveling costs, we arrive to j from exactly one i, we leave i to exactly one j.

Example – 5 towns – cycle and subcycles (subroute)





Kafka (2013)

Subroute elimination conditions I

- $x_{ii} = 0, \ c_{ii} = \infty$
- $x_{ij} + x_{ji} \le 1$
- $x_{ij} + x_{jk} + x_{ki} \le 2$
- ...
- $\sum_{i \in S} \sum_{j \in S} x_{ij} \le |S| - 1, \ S \subseteq \{1, \dots, n\}, \ 2 \le |S| \le n - 1$

Approximately 2^n inequalities, it is possible to reduce to $|S| \le \lceil n/2 \rceil$.

Subroute elimination conditions II

Other valid inequalities (using additional real decision variables u_i):

$$u_i - u_j + nx_{ij} \le n-1, i,j = 2,\dots,n.$$

Eliminate subroutes: There is at least one route which does not go through vertex 1, denote this route by C and the number of edges by $|E(C)|$. If we sum all these inequalities over all edges $\{i,j\}$, which are in C, i.e. the corresponding variables $x_{ij}=1$, we obtain

$$n|E(C)| \le (n-1)|E(C)|, \tag{5}$$

which is a contradiction.

Subroute elimination conditions II

$$u_i - u_j + nx_{ij} \le n-1, i,j = 2,\dots,n.$$

Hamilton cycle is feasible: let the vertices be ordered as $v_1 = 1$, v_2, ..., v_n. We set $u_i = l$, if $v_l = i$, i.e. u_i represent the order. For each edge of the cycle $\{i,j\}$ it holds $u_i - u_j = -1$, i.e.

$$u_i - u_j + nx_{ij} = -1 + n \le n-1. \tag{6}$$

For edges, which are not in the cycle, the inequality holds too: $u_i - u_j \le n-1$ and $x_{ij} = 0$.

Subroute elimination conditions – example

Consider subroutes: 1–4–5, 2–3

Add inequalities

$$\begin{aligned} u_2 - u_3 + 5x_{23} &\le 4, \\ u_3 - u_2 + 5x_{32} &\le 4, \end{aligned}$$

or

$$x_{23} + x_{32} \le 1.$$

TSP – computational complexity

$\mathcal{NP}$ (Nondeterministic Polynomial) is the class of decision problems with the property that: for any instance for which the answer is YES, there is a polynomial proof of the YES.

Traveling Salesman Problem with Time Windows

- t_i – time when customer i is visited
- T_{ij} – time necessary to reach j from i
- l_i, u_i – lower and upper bound (time window) for visiting customer i
- M – a large constant

$$
\begin{aligned}
\min_{x_{ij},t_i} \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & & & \quad (7)\\
\sum_{i=1}^{n} x_{ij} &=& 1,\ j=1,\dots,n, & \quad (8)\\
\sum_{j=1}^{n} x_{ij} &=& 1,\ i=1,\dots,n, & \quad (9)\\
t_i + T_{ij} - t_j &\le& M(1-x_{ij})\ i,j=1,\dots,n, & \quad (10)\\
l_i \le t_i &\le& u_i,\ i=1,\dots,n, & \quad (11)\\
x_{ij} &\in& \{0,1\}. &
\end{aligned}
$$

Capacitated Vehicle Routing Problem

Parameters

- n – number of customers
- 0 – depo (starting and finishing point of each vehicle)
- K – number of vehicles (homogeneous)
- $d_j \ge 0$ – customer demand, for depo $d_0 = 0$
- $Q > 0$ – vehicle capacity ($KQ \ge \sum_{j=1}^{n} d_j$)
- c_{ij} – transportation costs from i to j (usually $c_{ii} = 0$)

Decision variables

- x_{ij} – equal to 1, if j follows after i on the route, 0 otherwise
- u_j – upper bound on transported amount after visiting customer j

Capacitated Vehicle Routing Problem

$$
\begin{aligned}
\min_{x_{ij},u_i} \sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x_{ij} & & & \quad (12)\\
\sum_{i=0}^{n} x_{ij} &=& 1,\ j=1,\dots,n, & \quad (13)\\
\sum_{j=0}^{n} x_{ij} &=& 1,\ i=1,\dots,n, & \quad (14)\\
\sum_{i=1}^{n} x_{i0} &=& K, & \quad (15)\\
\sum_{j=1}^{n} x_{0j} &=& K, & \quad (16)\\
u_i - u_j + d_j &\le& Q(1-x_{ij})\ i,j=1,\dots,n, & \quad (17)\\
d_i \le u_i &\le& Q,\ i=1,\dots,n, & \quad (18)\\
x_{ij} &\in& \{0,1\}. &
\end{aligned}
$$

Capacitated Vehicle Routing Problem

(12) minimization of transportation costs
(13) exactly one vehicle arrives to customer j
(14) exactly one vehicle leaves customer i
(15) exactly K vehicles return to depot 0
(16) exactly K vehicles leave depot 0
(17) balance conditions of transported amount (serve also as subroute elimination conditions)
(18) bounds on the vehicle capacity

(All vehicles are employed.)

Greedy heuristic

Start with an empty set (solution) and choose the item with **the best immediate reward** at each step.

Example: Traveling Salesman Problem with the (symmetric) distance matrix

$$\begin{pmatrix} - & 9 & 2 & 8 & 12 & 11 \\ & - & 7 & 19 & 10 & 32 \\ & & - & 29 & 18 & 6 \\ & & & - & 24 & 3 \\ & & & & - & 19 \\ & & & & & - \end{pmatrix}$$

Greedy steps: 1–3 (2), 3–6 (6), 6–4 (3), 4–2 (19), 2–5 (10), 5–1 (12), i.e. the route length is 52.

Local search heuristic

Choose an initial solution x and search its neighborhood $U(x)$. Repeat until you are able to find a better solution, i.e. if $y \in U(x)$, $f(y) < f(x)$, set $x = y$.

Example: Traveling Salesman Problem, define the neighborhood $U(x)$ as **2-exchange**, i.e. if $S = \{(i,j) \in A : \; x_{ij} = 1\}$ is a feasible solution, then

$$U(x) = \{S' : \; |S \cap S'| = n - 2\},$$

in other words: **replace edges** (i,j), (i',j') by (i,i'), (j,j').

Greedy steps: 1–3 (2), 3–6 (6), 6–4 (3), 4–2 (19), 2–5 (10), 5–1 (12), i.e. the route length is 52.

2-exchange: 1–3 (2), 3–4 (29), 4–6 (3), 6–2 (32), 2–5 (10), 5–1 (12), i.e. the route length is 88.

Basic heuristics for VRP

Insertion heuristic:

- Start with empty routes.
- FOR all customers DO: Insert the customer to the place in a route where it causes the lowest increase of the traveled distance.

Clustering:

- Cluster the customers according to their geographic positions ("angles").
- Solve[1] the traveling salesman problem in each cluster.

Possible difficulties: time windows, vehicle capacities, ...

[1]..exactly, if the clusters are not large.

Tabu search for VRP

For a given number of iteration, run the following steps:

- Find the best solution in a *neighborhood* of the current solution. Such solution can be worse than the current one or even infeasible (use a penalty function).
- Forbid moving back for a random number of steps by actualizing the **tabu list**.
- Remember the best solution.

The tabu search algorithm enables moving from local solutions (compared with a simple "hill climbing alg.").

Genetic algorithms

Iterative procedure:

- Population – finite set of individuals with genes
- Generation
- Evaluation – fitness
- Parent selection
- Crossover produces one or two new solutions (offspring).
- Mutation
- Population selection

Rich Vehicle Routing Problems

- **Goal** – maximization of the ship *filling rate* (operational planning), optimization of fleet composition, i.e. number and capacity of the ships (strategic planning)
- **Rich Vehicle Routing Problem**
  - time windows
  - heterogeneous fleet (vehicles with different capacities and speed)
  - several depots with inter-depot trips
  - several routes during the planning horizon
  - *non-Euclidean distances* (fjords)
- Mixed-integer programming :-(, constructive heuristics for getting an initial feasible solution and tabu search

- M. Branda, K. Haugen, J. Novotný, A. Olstad, **Downstream logistics optimization at EWOS Norway**. Research report.

Rich Vehicle Routing Problems

Our approach

- Mathematical formulation
- GAMS implementation
- Heuristic (insertion heuristic, tabu search) implementation
- Decision Support System (DSS)

Literature

- M. Branda, J. Novotný, A. Olstad: **Fixed interval scheduling under uncertainty - a tabu search algorithm for an extended robust coloring formulation**. Computers & Industrial Engineering 93, 45–54.
- M. Branda, K. Haugen, J. Novotný, A. Olstad: **Downstream logistic optimization at EWOS Norway**. Mathematics for Applications 6 (2), 127-141.
- O. Kafka: **Optimální plánování rozvozu pomocí dopravních prostředků**, Diploma thesis MFF UK, 2013. (IN CZECH)
- P. Toth, D. Vigo (2002). **The vehicle routing problem**, SIAM, Philadelphia.
- L.A. Wolsey (1998). **Integer Programming**. Wiley, New York.
- L.A. Wolsey, G.L. Nemhauser (1999). **Integer and combinatorial optimization**. Wiley, New York.