Multi-vehicle routing: the VRP

Logistics

Giovanni Righini

Università degli Studi di Milano

The Vehicle Routing Problem

In its most basic version, the vehicle routing problem (VRP) is the following:

- a set of locations (nodes of a graph $G = (N, A)$) must be visited;
- distance/time/cost between them (arc weights) is known;
- a fleet of identical vehicles is available;
- all vehicles start from a common depot (node 0) and return to it;
- some additional constraints must be satisfied (capacity, time windows, maximum route length,...).

The typical objective to be minimized is the total distance traveled.

The VRP is NP-hard.

Several exact and heuristic algorithms have been devised to solve it.

2-index arc flow formulation

Binary variables are associated with arcs: $x_{ij} = 1$ if and only if arc (i,j) belongs to the solution.

$$
\begin{aligned}
\text{minimize } z = & \sum_{(i,j)\in A} w_{ij} x_{ij} & \\
\text{s.t. } & \sum_{j\in N} x_{ij} = 1 & \forall i \in N, i \neq 0 \\
& \sum_{i\in N} x_{ij} = 1 & \forall j \in N, j \neq 0 \\
& \sum_{i\in N} x_{i0} = \sum_{j\in N} x_{0j} \leq K & \\
& S.E.C. & \\
& additional constraints & \\
& x_{ij} \in \{0,1\} & \forall (i,j) \in A
\end{aligned}
$$

Sub-tour Elimination Constraints are needed to forbid sub-tours not including the depot.

Extended formulation

An extended formulation of the VRP is the following:

$$\begin{aligned}
\text{minimize } z = \sum_{r\in R} c_r \theta_r & & \\
\text{s.t. } \sum_{r\in R} a_i^r \theta_r \geq 1 & & \forall i \in N \\
\sum_{r\in R} \theta_r \leq K & & \\
\theta_r \in \{0,1\} & & \forall r \in R
\end{aligned}$$

where R is the (exponentially large) set of feasible routes.

This model has only $n+1$ constraints but exponentially many variables.

It is solved by branch(-and-cut)-and-price.

Reduced costs

$$\begin{aligned}
\text{minimize } z = \sum_{r \in R} c_r \theta_r & & \\
\text{s.t. } \sum_{r \in R} a_i^r \theta_r \geq 1 & \qquad \forall i \in N & \qquad (1)\\
\sum_{r \in R} \theta_r \leq K & & \qquad (2)\\
\theta_r \in \{0, 1\} & \qquad \forall r \in R. &
\end{aligned}$$

$\lambda \geq 0$: dual variables of covering constraints 1
$\mu \leq 0$: dual variable of the convexity constraint 2.

Reduced cost:

$$\overline{c}_r = c_r - \sum_{i \in N} \lambda_i a_i^r - \mu \;\; \forall r \in R.$$

The pricing sub-problem

The route cost is $c_r = \sum_{(i,j)\in A} w_{ij}x_{ij}$.

The binary coefficients are $a_i^r = \sum_{(i,j)\in A} x_{ij}$.

The pricing sub-problem is:

$$
\begin{aligned}
\text{minimize } \overline{c} = & \sum_{(i,j)\in A} w_{ij}x_{ij} - \sum_{i\in N} \lambda_i a_i - \mu \\
\text{s.t. } & \sum_{(i,j)\in A} x_{ij} = \sum_{(i,j)\in A} x_{ij} = a_i && \forall i \in N \\
& a_0 = 1 \\
& a_i \leq 1 && \forall i \in N, i \neq 0 \\
& \text{S.E.C.} \\
& \text{additional constraints} \\
& x_{ij} \in \{0,1\} && \forall (i,j) \in A.
\end{aligned}
$$

A graph with negative cycles

The objective function

$$\overline{c} = \sum_{(i,j)\in A} w_{ij} x_{ij} - \sum_{i\in N} \lambda_i a_i - \mu$$

can be rewritten as

$$\overline{c} = \sum_{(i,j)\in A} \overline{w}_{ij} x_{ij}$$

where

- μ is a constant that has no effect on the optimization;
- $\overline{w}_{ij} = w_{ij} - \lambda_i$ for all arcs.

S.E.C. would not be required fo find a shortest path on a graph with no negative cycles.

Owing to the effect of λ, negative cost cycles can occur.

Pricing sub-problem complexity

Pricing sub-problem: Elementary Shortest Path Problem (ESPP).

The problem in *NP*-hard.

Hence

- the linear relaxation lower bound provided by the extended formulation can be tighter than the linear relaxation lower bound provided by the compact formulation;
- we need an effective algorithm to solve the ESPP.

Dynamic programming

State: (i, S), where

- i is the last reached node;
- S is the subset of visited nodes.

Extension from state (i, S) to (j, S'):

- Condition: $j \notin S$
- $S' \leftarrow S \cup \{j\}$
- $c(j, S') = c(i, S) + \overline{w}_{ij}$

Dominance: (i, S') dominates (i, S'') when

- $S' \subset S''$
- $c(i, S') \leq c(i, S'')$

and at least one of the inequalities is strict.

Resources

Besides S, it may be necessary to consider additional resources (RCESPP), such as

- the amount of capacity already consumed,
- the total time elapsed,
- the total energy/fuel consumed,
- the total distance traveled,
- et cetera...

to impose additional constraints.

State: (i, S, q)
Extension condition: $q + h_{ij} \leq Q$, where Q is the total amount of available resource and h_{ij} is the resource consumption when traveling on (i, j).
Update: $q' \leftarrow q + h_{ij}$.
Dominance: $q' \leq q''$.

Combinatorial explosion

Two main techniques to cope with the combinatorial explosion in the number of states:

- accept non-elementary routes: the master problem includes infeasible columns (routes with cycles) and it provides looser lower bounds;
- speed-up the computation of feasible (cycle-free) routes:
  - bi-directional search;
  - state-space relaxation.

Bi-directional search

The number of D.P. states grows much faster than linearly in the number of nodes in the path.

Finding all non-dominated paths of length $n/2$ takes much less than half the time needed to find all non-dominated paths of length n.

Since origin and destination of the route are both known (the depot in both cases), than each feasible route of length at most n can be split into two parts of length at most $n/2$.

For this we need:

- backward extension rules, dominance rules etc. if the graph is oriented;
- an extension stop rule;
- a procedure to efficiently join pairs of semi-routes into routes.

Extension stop rule

Extension of states can be terminated as soon as half of the total available amount of a (suitable chosen) critical resource has been consumed.

We have the guarantee that no route is missed if all semi-routes are generated up to this half-way point.

When there are several resources involved, selecting the critical resource may be non-trivial.

Join procedure

- Symmetric graph: a single set of semi-routes;
- Asymmetric graph: two distinct sets of forward and backward semi-routes.

The sets can be sorted (e.g. according to the reduced cost of the semi-routes).

The best semi-routes in each set are likely to produce infeasible routes.

The worst semi-routes in each set are likely to produce positive reduced cost routes.

Heuristic search algorithms are needed to quickly find good pairs of semi-routes (possibly, many of them).

State space relaxation

The need of storing S, the subset of already visited nodes, produces a combinatorial explosion in the number of states.

State space relaxation: replace S with $\sigma = |S|$ (fewer states).

Condition for extension: $\sigma < n$.
Dominance test: $\sigma' < \sigma''$ (more likely to succeed).

Some information is lost.
Routes with more than n nodes can still be forbidden (no infinite loops).
Cycles can occur in the optimal solution.

D.P. provides a lower bound, instead of the optimum.

Decremental SSR

DSSR: Consider a subset $V \subseteq N$ of critical nodes (initially empty).

State: subset $\overline{S}$ of visited critical nodes.

Condition for extension from i to j: $\sigma < n$ and $j \notin \overline{S}$.
Dominance test: $\sigma' < \sigma''$ and $\overline{S}' \subseteq \overline{S}''$.

Information on critical nodes is no longer lost.
Only non-critical nodes can be visited more than once.
Cycles can still occur in the optimal solution.
If this happens, they are inserted into the subset of critical vertices and the procedure is repeated.

D.P. with DSSR provides a valid lower bound at each iteration and eventually it computes the optimum.
The computing time increases with $|\overline{S}|$.
Experimentally, a rather small fraction of nodes is needed to reach the optimum.