Dynamic Vehicle Routing Problem Definition

Michał Okulewicz

March 23, 2015

1 Definition

In this class of Dynamic Vehicle Routing Problem[1] one considers:

- a fleet V of n vehicles,
- a series C of m clients (requests) to be served,
- a set D of k depots from which vehicles may start their routes.

The fleet V is homogeneous, i.e. vehicles have identical *capacity* $\in \mathbb{R}$ and the same *speed*[2] $\in \mathbb{R}$. The cargo is taken from one of the k depots[3]. Each depot $d_j \in D, j = 1, \ldots, k$ has:

- a certain location $location_j \in \mathbb{R}^2$,
- working hours $(start_j, end_j)$, where $0 \leq start_j < end_j$.

For the sake of simplicity, we additionally define two global auxiliary variables (constraints): $start := \min\limits_{j \in 1,\ldots,k} start_j$ and $end := \max\limits_{j \in 1,\ldots,k} end_j$, which are not part of the standard definition.

Each client $c_l \in C$ $(l = k+1, \ldots, k+m)$, has a given:

- $location_l \in \mathbb{R}^2$,
- $time_l \in \mathbb{R}$, which is a point of time when their request becomes available $(start \leq time_l \leq end)$,

[1]The definition comes from the article (in preparation) *A 2-Phase PSO Algorithm for Solving Dynamic Vehicle Routing Problem* by M. Okulewicz and J. Mańdziuk

[2]In all benchmarks used in this paper *speed* is defined as one distance unit per one time unit.

[3]In the most common benchmarks used in the literature, $k = 1$.

- $unld_l \in \mathbb{R}$, which is the time required to unload the cargo,
- $size_l \in \mathbb{R}$ - size of the request ($size_l \leq capacity$).

A travel distance $\rho(i,j)$ is an Euclidean distance between $location_i$ and $location_j$ on the $\mathbb{R}^2$ plane, $i,j = 1,\ldots,m+k$.

For each vehicle v_i, the $r_i = (i_1, i_2, \ldots, i_{p(i)})$ is a permutation of indexes of requests assigned to the ith vehicle and depots to be visited by the vehicle, which defines the route of the ith vehicle. Please note, that the first and the last elements always denote depots (the initial one and the final one, respectively). The arv_{i_j} is the time of arrival to the jth request of the ith vehicle (induced by the permutation r_i and the time when requests become available - see eq. (2) and (3)).

As previously stated, the goal is to serve the clients (requests), according to their defined constraints, with minimal total cost (travel distance) within the time constraints imposed by the opening hours of the depot.

In other words, the goal of the algorithm is to find such a set R of permutations $R = \{r_1^*, r_2^*, \ldots, r_n^*\}$ of requests and depots that minimizes the following cost function:

$$COST(r_1, r_2, \ldots, r_n) = \sum_{i=1}^{n} \sum_{j=2}^{p(i)} \rho(i_j, i_{j-1}) \tag{1}$$

under the following constraints (2) - (6).

Vehicle $i, i = 1, 2, \ldots, n$ cannot arrive at $location_{i_j}$ before the time required for traveling from the last visited location $location_{i_{j-1}}$ (which could be the depot from which it starts) after receiving an information about the new request, is completed:

$$\begin{aligned} \forall_{i\in\{1,2,\ldots,n\}} \forall_{j\in r_i/\{i_1\}} \quad & arv_{i_j} \\ & \geq time_{i_j} + \rho(i_j, i_{j-1}) \end{aligned} \tag{2}$$

The vehicle cannot arrive at $location_{i_j}$ before serving the request $r_{i_{j-1}}$ and traveling to the next location:

$$\begin{aligned} \forall_{i\in\{1,2,\ldots,n\}} \forall_{j\in r_i/\{i_1\}} \quad & arv_{i_j} \\ \geq arv_{i_{j-1}} + unld_{i_{j-1}} & + \rho(i_j, i_{j-1}) \end{aligned} \tag{3}$$

All vehicles must return to the depot before its closing:

$$\forall_{i\in\{1,2,\ldots,n\}} \quad arv_{i_{p(i)}} \leq end_{i_{p(i)}} \tag{4}$$

A sum of requests' sizes between consecutive visits to the depots must not exceed vehicle's capacity:

$$\forall_{i\in\{1,2,\ldots,n\}}\forall_{j_1,j_2\in r_i, j_1<j_2, i_{j_1}\leq k}: \quad \sum_{l=j_1}^{j_2} size_{i_l} \\ \leq capacity \cdot |\{l' : i_{l'} \leq k \wedge l' \in [j_1, j_2-1]\}| \tag{5}$$

Each client must be assigned to exactly one vehicle:

$$\forall_{l\in\{1,2,\ldots,m\}}\exists!_{i\in\{1,2,\ldots n\}} \quad (l+k)\in r_i \tag{6}$$

2 Cut-off time parameter

The important parameter for the DVRP benchmarks is setting the *cut-off* time parameter. In the real business situation this could be interpreted as not accepting any new requests after the *cut-off* time (given as part of the working day - from opening to closing of the depot) and moving them as known request to the next working day. As we are here only solving the benchmarks we set the value of $time_l = 0$ if $time_l > T_{\text{cut-off}}(\max_{j\in 1,\ldots,k} end_j)$. This parameter should be set to $T_{\text{cut-off}} = 0.5$, but should remain a parameter of the problem.

3 Important links

- Benchmark files without description
- Benchmark files with description
- Small benchmark files - you should be able to compute them in around 5 to 240 minutes (depending on their size)
- My webpage for MSI2 - you could find there more papers concerning DVRP, so you could get more intuition about the problem, best known dynamic results and role of cut-off time parameter - but **not the proper algorithm**.
- Visualisation of some of the best known results.
- Different VRP versions - you could find references to different types of static Vehicle Routing Problem, which might be useful in designing your algorithm.

4 Your task

You should develop an efficient distributed/parallel version of the exact (brute-force with tricks - if you prefer) algorithm for solving the DVRP with **full knowledge** of the time of appearance of requests but with maintaining the constraints (2) and (3).

Hint This problem could be considered as Traveling Salesman Problem problem within the Set Partitioning Problem.