A TIME LIMITED VEHICLE ROUTING PROBLEM

A vehicle routing problem is a classical problem in operation research consisting in delivery routes optimization in communications network containing depot of all routes and a given number of cities, which is necessary to include in delivery routes. The demand of all cities is given and the condition is that the sum of demands of the cities on the route should be less or equal to the capacity of a vehicle.

The paper deals with a modification of the vehicle routing problem in which the capacity of a vehicle is not limited but the times of pick-up to all cities are limited by a given value. A similar problem and model can be formulated for optimization of pick-up routes with pick-up time limited. The mathematical model proposed for the problem is based on Miller-Tucker-Zemlin formulation of the traveling salesman problem. As the time limited vehicle routing problem is NP hard, a solution to huge problems cannot be obtained in acceptable computation time. The following heuristics are proposed for the time limited traveling salesman problem: nearest neighborhood method, insert method and savings method. Use of all methods is illustrated by numerical experiment. The difference between results obtained by those methods is shown on the case study.

1. Introduction and a mathematical model

This modification of vehicle routing problem (or the traveling salesman problem) is based on the bank payment orders pick-up case study. The depot (called the centre) is the city, to which daily payment orders are transported from the bank branches. The payment orders have to be delivered to the centre in a given time, i.e. the time interval between loading payment orders at the branches and unloading them at the centre must not exceed a given value. Vehicle capacity is not the limiting factor in this problem. The transportation is executed at the public road network. The shortest distances between cities, or between the cities and the centre, are known.

Let us have \( n \) cities connected by the road network with the shortest distance matrix \( C \). Let us suppose that the city 1 is the centre to which the payment orders are transported from other \( n-1 \) cities. The pick-up from each city must be finished in the time \( T \), which forecloses the pick-up payment orders by one Hamiltonian cycle. The result forms more routes-cycles; each of them includes city 1 (as in the vehicle routing problem). The pick-up time \( T \) starts to be computed since the departure time from the first branch on the route. Thus, the traveling time from the city 1 to the first branch is not included in the time \( T \). For the model of the problem we would need the traveling times between each pair of cities. Because of simplicity, these times are derived from the vehicle average speed and distances given by matrix \( C \). Instead of a time limited traveling salesman problem: nearest neighborhood method, insert method and savings method. Use of all methods is illustrated by numerical experiment. The difference between results obtained by those methods is shown on the case study.

The time limited vehicle routing problem (TLVRP) is similarly as the traveling salesman problem (which is reduced on the vehicle routing problem in case of a big enough value \( L \)) NP hard.

Let us set the mathematical model of TLVRP first (for delivery case of TLVRP). The binary variable \( x_{ij} \) equals 1, if the edge \((i, j)\) is included in the route, i.e. the vehicle goes from city \( i \) to city \( j \); otherwise this variable equals zero.

The mathematical model of TLVRP is:

\[
\begin{align*}
\min & \quad \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \\
\text{subject to} & \quad \sum_{j=1}^n x_{ij} = 1, \quad i = 1, 2, \ldots, n, \quad (2) \\
& \quad \sum_{i=1}^n x_{ij} = 1, \quad i = 1, 2, \ldots, n, \quad (3) \\
& \quad u_i + c_{ij} - M(1 - x_{ij}) \leq u_j, \quad i = 1, 2, \ldots, n, \quad j = 2, 3, \ldots, n, \quad i \neq j, \quad (4) \\
& \quad u_j \leq L, \quad i = 1, 2, \ldots, n, \quad (5) \\
& \quad x_{ij} \in [0,1], \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, n, \quad i \neq j, \quad (6)
\end{align*}
\]

The objective function (1) represents the total length of all the routes. The set of equations (2) assures that only one edge comes out from each city. Similarly, constraints (3) state that only one edge comes to each city. Equations (2) and (3) are not applied on city 1, because from and to city 1 there come as many edges as there are routes. The set of inequalities (4) defines variable \( u_j \) corresponding to the length of the route from city 1 to city \( i \). The con-
ditions (5) mean that no such length exceeds the limit \( L \). The constant \( M \) is a big enough positive number (or \( L + c_{y} - c_{l} \)).

2. Heuristic methods

As the TLVRP belongs to the class of NP hard problems, it will be useful to propose heuristic methods for a large scale problem solution. The heuristic methods used for the traveling salesman problem or the vehicle routing problem will be modified. It is assumed that the matrix \( C \) is symmetric and nonnegative, and \( c_{ij} = L \) for \( i = 2, 3, \ldots, n \) (otherwise, the TLVRP does not have a feasible solution).

The following notation will be used in the proposed modifications of heuristic methods. Let us denote by \( M \) the set of cities that were not included in any route. At the beginning of the method set \( M \) equals \( \{ 2, 3, \ldots, n \} \). The heuristic method ends when the set \( M \) is empty. The route will be denoted as \( tr = (tr(1), tr(2), \ldots, tr(m)) \), where \( tr(1) = tr(m) = 1 \). The constraints (5) will be tested for each possible change of the route (enlargement) in the form:

\[
P_1: \sum_{l=1}^{m-2} c_{tr(l), tr(l+1)} \leq L, \quad \text{or} \quad P_2: \sum_{l=2}^{m-1} c_{tr(l), tr(l+1)} \leq L.
\]

If the condition \( P_1 \) is satisfied, the route is feasible for delivery (in the case of pick-up the \( tr \) should be reversed). If the condition \( P_2 \) is satisfied, the route is feasible for the pick-up; for delivery it is necessary to reverse \( tr \).

2.1 The nearest neighborhood method

In this method the following steps are executed until the set \( M \) is empty:

Step 1. Let us denote the city with the shortest distance \( c_{ij} \) as \( k \) and let it form the route \( tr(1) = 1, tr(2) = k, tr(3) = 1 \), let \( m = 3 \). City \( k \) is deleted from the set \( M \).

Step 2. Let us find city \( k \) from \( M \) which minimizes the distance \( c_{tr(m-1),k} \) and for which the route enlargement \( tr \) when inserting this city after city \( tr(m-1) \) satisfies the condition \( P_1 \) or \( P_2 \). If such a city \( k \) does not exist, the route is closed and the method follows by step 1 (starting the new route).

Step 3. Let us enlarge the route \( tr \) by inserting city \( k \) after the city \( tr(m-1) \), let \( m \) increase by 1 and delete city \( k \) from set \( M \). If set \( M \) is not empty, follow step 2; otherwise, the method ends.

2.2 The insert method

The following steps are taken until set \( M \) is empty.

Step 1. Let us denote city \( k \) with the greatest distance \( c_{ij} \) and put the route \( tr(1) = 1, tr(2) = k, tr(3) = 1 \), and \( m = 3 \). City \( k \) is deleted from set \( M \). If set \( M \) is empty the method ends.

Step 2. City \( k \) from set \( M \) being found minimizes value \( d = c_{tr(i),k} + c_{k,tr(i+1)} - c_{tr(i),tr(i+1)} \) for all \( i = 1, 2, \ldots, m-1 \) and \( k \in M \). The route \( tr \) satisfies the condition \( P_1 \) or \( P_2 \) if this route is enlarged by inserting city \( k \) between cities \( tr(i) \) and \( tr(i+1) \), where \( i \) minimizes the value \( d \). If city \( k \) does not exist, the route is closed and the method follows by step 1 (start a new route). Enlarge the route \( tr \) by inserting city \( k \) between the city \( tr(i) \) and \( tr(i+1) \), increase \( m \) by 1. City \( k \) will be deleted from set \( M \). If set \( M \) is empty the method ends; otherwise, it follows by step 2.

2.3 The savings method

The following steps are executed until set \( M \) is empty.

Step 1. If set \( M \) contains only city \( k \), the route \( tr(1) = 1, tr(2) = k, tr(3) = 1 \) is formed; let \( m = 3 \). City \( k \) will be deleted from set \( M \) and the method ends.

Step 2. Let it find the pair of cities from \( M \) in the form \((k, l)\), that satisfies the condition and maximizes savings. If this pair \((k, l)\) does not exist, form direct routes \((1, k), (k, 1)\) for all each city \( k \) from \( M \), stop.

Step 3. Put the route \( tr(1) = 1, tr(2) = k, tr(3) = l, tr(4) = 1, m = 4 \).

Step 4. Find city \( i \) from \( M \) that maximizes \( s_{ik} \), or city \( j \) from \( M \) that maximizes \( s_{ij} \), such that the route, being formed by inserting \( i \) into the route before city \( k \), satisfies conditions \( P_1 \) or \( P_2 \), or being formed by inserting \( j \) into the route after city \( l \), satisfies conditions \( P_1 \) or \( P_2 \). If either \( i \) or \( j \) do not exist, the route is closed and the method follows by step 1.

Step 3. If \((or \ j \) does not exist), city \( i \) is inserted into the route before city \( k \) and city \( i \) is deleted from set \( M \). Let \( s \) increase by 1 and \( k \) equal \( l \). If \((or \ i \) does not exist), city \( j \) is inserted into the route after city \( l \) and city \( j \) is deleted from \( M \). Let \( m \) increase by 1. The method follows by step 4.

3. Numerical experiments

The proposed model and heuristics were applied to a set of problems and the results are shown in table 1. It contains 20 time limited vehicle routing problems, which are solved by using heuristics proposed above, by the model (1) – (6) and LINGO8.0 solver and CPLEX9.0 solver. The value of objective function obtained using heuristics (the best value of three heuristic methods) and model are shown. Beside the objective value, the computation time and lower bound of the objective function are placed in table 1. The computation time limit was 1 hour.
### Computation experiments: results

| # nodes | heuristics | LINGO | CPLEX |
|---------|------------|-------|-------|
|         |            | lower bound | comp. time | optimal model | lower bound | comp. time | optimal model | lower bound |
| 1       | 26         | 439.914 508  | 346 1 hour | no 418 347   |           |           |           |           |
| 2       | 19         | 345.36 324.36 | — 3 mins yes | 324.36 — |           |           |           |           |
| 3       | 12         | 271.097 271.097 | — 1 min yes | 271.097 — |           |           |           |           |
| 4       | 29         | 514.598 471.071 | 424 1 hour no | 491 417   |           |           |           |           |
| 5       | 14         | 261.402 253.283 | — 1 min yes | 253.283 — |           |           |           |           |
| 6       | 20         | 370.395 345.079 | 323 1 hour no | 345.079 345.079 |           |           |           |           |
| 7       | 22         | 501.555 493.55 | 390 1 hour no | 473 392   |           |           |           |           |
| 8       | 15         | 286.684 286.68 | — 1 min yes | 286.68 — |           |           |           |           |
| 9       | 25         | 346.494 320.58 | — 14 mins yes | 320.58 — |           |           |           |           |
| 10      | 11         | 157.64 154.315 | — 1 min yes | 154.315 — |           |           |           |           |
| 11      | 26         | 319.074 367.6 | 300.4 1 hour no | 340 301   |           |           |           |           |
| 12      | 32         | 619.617 837 | 450 1 hour no | 650 462   |           |           |           |           |
| 13      | 18         | 272.205 272.2 | — 25 mins yes | 272.2 — |           |           |           |           |
| 14      | 20         | 356.7 322.69 | 287 1 hour no | 322.69 298 |           |           |           |           |
| 15      | 27         | 383.177 442 | 307 1 hour no | 361 307   |           |           |           |           |
| 16      | 53         | 830.402 990.8 | 573 1 hour no | 868 573   |           |           |           |           |
| 17      | 16         | 235.833 233.6 | — 25 mins yes | 233.6 — |           |           |           |           |
| 18      | 27         | 434.497 473.7 | 334 1 hour no | 424 337   |           |           |           |           |
| 19      | 32         | 619.617 837 | 450 1 hour no | 650 462   |           |           |           |           |
| 20      | 16         | 219.477 204.67 | — 1 min yes | 204.67 — |           |           |           |           |

### References

[1] GUTIN, G., PUNNEN, A. P.: *The traveling salesman problem and its variations*. Kluver 2002. ISBN 1-4020-0664-0