1. Introduction

Messenger problem, as a variation of pickup and delivery problem, deals with the transport of a set of packages from their origins to given destinations. In reality, several vehicles have to be used to be able to satisfy all requirements within given time limit. Messengers can be located in one or multiple depots. Because of NP-hardness of the problem it is impossible, for most real problems, to find the optimal solution in acceptable time. Therefore, heuristic algorithms must be used. In the paper, insertion method for routes generation is presented. As the computational experiments in VBA for Excel show, obtained results can be improved using the exchange algorithm.

Keywords: Multiple messenger problem, insertion method, exchange algorithm.

2. Insertion method for multiple messenger problem with multiple depots

In messenger problem, a location of pickup and a location of delivery are given for each shipment. Suppose $n$ packages have to be transported by messengers, all are known in advance. Let us denote all pickup nodes in the network by the even numbers and all delivery nodes by the odd numbers. In addition, if the package is picked up at node $i$, it will be delivered at node $i + 1$. For service, $K$ messengers in separated depots are available. Let us assume, because of the proposed algorithms, $K$ is an odd number. In case the real number of messengers is even, a dummy messenger is introduced. Nodes $k = 1, 2, ..., K$ specify depots, nodes $K + 1, K + 3, ..., K + 2n - 1$ correspond to pickup locations and $K + 2, K + 4, ..., K + 2n$ denote delivery locations. Minimal distances $c_{ij}$ and associated travel times $d_{ij}$ for all pairs of locations are given ($i, j = 1, 2, ..., K + 2n; i \neq j$). In addition, $T_{max}$ is defined as the acceptable time for each route duration. In the following algorithm we suppose that all vehicles will be used for shipments delivery, i.e. $K$ routes will be generated. Let $U_i^k$ be a sequence of $h_i$ nodes included in route $k$ of length $z_k$ and duration $T_k$, and $U_j^k$ be a set of nodes that can be still included in route $k$ (in the certain step of routes generation). Number $u_i \in U_i^k$ corresponds to the node on position $i$ in sequence $U_i^k$. Value of $z$ is the total length of all generated routes.

The algorithm is based on inserting locations to current routes which extends their length. If insertion method is used for travelling salesperson problem, the route is obviously initialized.
by finding the farthest location \( s \) from the depot 1. Then, the cycle \([1, s, 1]\) is created as the initial route. Another location is inserted to the cycle with the objective to minimize its extension. This approach is repeated until all locations are incorporated in the Hamiltonian cycle. In case of multiple messenger problem, two important rules have to be respected: pickup location and related delivery location have to be inserted to the same route, and pickup location has to be inserted to the route in front of related delivery location. In addition, delivery location can be inserted exactly behind related pick-up one or behind several locations farther [8]. If multiple messengers are considered in the problem, finding the farthest location to each depot in the initialization step can be disadvantageous because that location can be very close to another depot. In such situation it would be inefficient to include the location in the route of the concerned messenger. In the proposed algorithm this weakness is eliminated. For each messenger \( k \), we are searching for the location that is closer to depot \( k \) than to any of remaining depots, as the initial location, it is taken the one with the minimal difference between its distance from depot \( k \) and minimal distance from other depots (see variables \( \Delta \)LowDist and \( \Delta \)LowLoc). If no such location exists, i.e. all locations are farther to depot \( k \) than to any of remaining depots, as the initial location, it is taken the one with the minimal difference between its distance from depot \( k \) and minimal distance from other depots (see variables \( Up\)Diff and \( Up\)Loc). Because at the beginning each messenger can visit all locations, all sets \( U_k \) include all pickup and delivery nodes.

**Step 1**

For \( k = 1, 2, \ldots, K \) set \( U_1 = \{K + 2, K + 4, \ldots, K + 2n\} \).

**Step 2**

For \( k = 1, 2, \ldots, K \) repeat

1. \( Low\)\text{Dist} = -\infty; \( Up\)\text{Diff} = +\infty; for all \( i \in U_k \) repeat

   if \( c_{il} \leq \min_{j \neq l} c_{ij} \) then

   if \( c_{il} > \text{LowDist} \) then

   \( \text{LowDist} = c_{il}; \text{LowLoc} = i \);

   else \( \left( c_{il} - \min_{j \neq l} c_{ij} \right) < \text{UpDiff} \) then

   \( \text{UpDiff} = \left( c_{il} - \min_{j \neq l} c_{ij} \right); \text{UpLoc} = i \);

   if \( \text{LowDist} > -\infty \) then \( s = \text{LowLoc} \) else \( s = \text{UpLoc} \);

   if \( s \) is even then

   \( U_1 = [k, s + 1, k]; z_s = c_{ks} + c_{s1} + c_{1k} \);

   \( t = t_{ks} + t_{s1} + t_{1k} \);

   else

   \( U_1 = [k, s - 1, k]; z_s = c_{ks} + c_{s1} + c_{1k} \);

   \( t = t_{ks} + t_{s1} + t_{1k} \);

   \( \Delta \text{Imm} = \min_{i=1,2, \ldots , K} \left( T_{ik} + T_{k+1, i} + T_{i, k+2} - t_{ik} \right); \)

   \( \Delta \text{Lat} = \min_{i=1,2, \ldots , K} \left( c_{ik} + c_{i1} + c_{1k} + c_{k+1, i} + c_{i, k+2} + c_{k+2, i} \right) \);\}

   \( \Delta \text{Loct} = \min_{i=1,2, \ldots , K} \left( c_{ik} + c_{i1} + c_{1k} + c_{k+1, i} + c_{i, k+2} + c_{k+2, i} \right) \);

   \( T = T_{s1} + T_{1k} + t_{ik} \);

   if \( k = 1, 2, \ldots, K \) repeat

   for all even \( r \in U_k \) repeat

   \( \Delta \text{Imm} = \min_{i=1,2, \ldots , K} \left( T_{ir} + T_{r+1, i} + T_{i, r+2} - t_{ir} \right); \)

   \( \Delta \text{Lat} = \min_{i=1,2, \ldots , K} \left( c_{ir} + c_{i1} + c_{1r} + c_{r+1, i} + c_{i, r+2} + c_{r+2, i} \right) \);

   \( \Delta \text{Loct} = \min_{i=1,2, \ldots , K} \left( c_{ir} + c_{i1} + c_{1r} + c_{r+1, i} + c_{i, r+2} + c_{r+2, i} \right) \);

   \( T = T_{r1} + T_{1r} + t_{ir} \);

   if \( k = 1, 2, \ldots, K \) repeat

   if \( s \) is even then \( U_1 = [s + 1, s - 1] \) else \( U_1 = [s - 1, s + 1] \).

**Step 3**

\( \Delta z = +\infty; \)

For \( k = 1, 2, \ldots, K \) repeat

for all even \( r \in U_k \) repeat

\( \Delta \text{Imm} = \min_{i=1,2, \ldots , K} \left( T_{ir} + T_{r+1, i} + T_{i, r+2} - t_{ir} \right); \)

\( \Delta \text{Lat} = \min_{i=1,2, \ldots , K} \left( c_{ir} + c_{i1} + c_{1r} + c_{r+1, i} + c_{i, r+2} + c_{r+2, i} \right) \);

\( T = T_{r1} + T_{1r} + t_{ir} \);

if \( k = 1, 2, \ldots, K \) repeat

for all \( r \in U_k \) repeat

\( \Delta \text{Imm} = \min_{i=1,2, \ldots , K} \left( T_{ir} + T_{r+1, i} + T_{i, r+2} - t_{ir} \right); \)

\( \Delta \text{Lat} = \min_{i=1,2, \ldots , K} \left( c_{ir} + c_{i1} + c_{1r} + c_{r+1, i} + c_{i, r+2} + c_{r+2, i} \right) \);

\( T = T_{r1} + T_{1r} + t_{ir} \);

if \( k = 1, 2, \ldots, K \) repeat

if \( \Delta \text{Imm} = \Delta \text{Lat} = +\infty \) then

insert nodes \( \text{Pick} \) and \( \text{Pick} + 1 \) to sequence \( U_{\text{seq}} \) to positions \( \text{After} 1 + 1 \) and \( \text{After} 1 + 2 \); else if \( \Delta z = +\infty \) then

insert nodes \( \text{Pick} \) and \( \text{Pick} + 1 \) to sequence \( U_{\text{seq}} \) to positions \( \text{After} 1 + 1 \) and \( \text{After} 2 + 1 \); if \( \Delta z = +\infty \) then

\( h_{\text{seq}} = h_{\text{seq}} + 2; z_{\text{seq}} = z_{\text{seq}} + \Delta z; T_{\text{seq}} = T_{\text{seq}} + \Delta T \);

for \( k = 1, 2, \ldots, K \) repeat

\( U_1 = U_1 \cup \{\text{Pick}\}; U_1 = U_1 \cup \{\text{Pick} + 1\} \).

**Step 4**

If \( U_1 = \emptyset \) for all \( k = 1, 2, \ldots, K \) then go to Step 5 else go to Step 3.

**Step 5**

\( z = \sum_{s=1}^{k} z_s \).

End.

In each iteration, the related pickup and delivery locations are inserted in two possible ways [8]: the delivery location will be visited immediately after visiting the pickup location, or it will be visited later, after visiting other locations. In the first case, value \( \Delta \text{Imm} \) expresses the minimal extension of the current route of messenger \( k \), if pickup node \( r \) and delivery node \( r + 1 \) are inserted together behind the node on position \( j \) in sequence \( U_1 \). Value \( \Delta \text{Imm} \) is calculated to check whether it is acceptable to include those locations in the route due to time limitation.
Similarly, in the second case, value \( \Delta z_{Lat,m} \) corresponds to the minimal extension of the route, if nodes \( r \) and \( r+1 \) are inserted to sequence \( U^*_i \) separately: node \( r \) on position \( v \) and node \( r+1 \) on position \( w \). Variable \( \Delta z_{Lat} \) is introduced to check feasibility in terms of time limitation.

After finding all acceptable insertions minimizing the extensions of individual routes, the most advantageous insertion of nodes \( \text{Pick} \) and \( \text{Pick} + 1 \) is found using the approach described in Step 3 of insertion method. For each route its minimal extension \( z_{Ext}(k = 1, 2, ..., K) \) is calculated. Their minimal value is selected:

\[
\min_{k=1}^{K} z_{Ext}(k) = z_{Ext,p}
\]

Let this extension of route \( p \) correspond to the insertion of node \( r \) behind node \( u_i \) and node \( r+1 \) behind node \( u_j \).

Step 3

If \( z_{Excl} = z_{Excl,p} \) then the exchange is realized, i.e. routes \( m \) and \( p \) are modified, and it is continued with Step 1, else with Step 4.

Step 4

End.

It should be emphasized that the proposed algorithm does not exchange shipments but transfers them from one route to another. Because in Step 2 time limitation \( T_{max} \) has to be respected, real exchange could be executed in case of infeasibility. However, it is necessary to check feasibility of both routes \( m \) and \( p \). The algorithm can be modified: all pairs of shipments across all routes are tested to feasibility and optimality.

If the algorithm finishes because the transfer cannot be realized (in Step 3), we can continue with Step 1 so that route \( m \) is added into “tabu list” and will be ignored when finding maximal reduction \( \Delta z_{Excl,m} \). Of course, it is possible to add other routes into “tabu list” if necessary.

4. Computational experiments

In [9] the application in VBA for Excel was developed on the base of proposed method and computational experiments were executed. They related the real matrix of minimal distances between 650 locations in the Czech Republic. Instances with 10, 20, 30 and 40 shipments were generated randomly for 3, 5, 7 and 11 messengers. In each category, 10 samples were examined (see Table 1).

### Table 1

| Depots | Shipments | IN x NN | NN Ex | Dist NN | IN Ex | Dist IN | IN Ex x NN Ex |
|--------|-----------|---------|-------|---------|-------|---------|---------------|
| 3      | 10        | 90.0    | 80.0  | 2.1     | 30.0  | 0.5     | 70.0          |
| 5      | 20        | 60.0    | 60.0  | 1.3     | 10.0  | 0.2     | 50.0          |
| 7      | 30        | 80.0    | 50.0  | 1.2     | 20.0  | 0.3     | 70.0          |
| 11     | 40        | 60.0    | 70.0  | 1.8     | 10.0  | 0.2     | 60.0          |
|        |           | 72.5    | 65.0  | 1.6     | 17.5  | 0.3     | 62.5          |
is not so expressive, final results show its higher efficiency in comparison with nearest neighbour algorithm. The statistically significant generalization of interesting conclusions requires the stronger analysis and wider experiments on real data. Greater savings would be probably achieved by the modifications mentioned above. Similarly to other special vehicle routing problems [10], time windows can be considered in multiple messenger problem. The proposed heuristic methods can be also extended to multiple messenger problems with a limited capacity of vehicles. As it was proved on the single vehicle messenger problem [8], insertion algorithm has its significance especially in case of dynamic problems. Future research will be aimed at using heuristic algorithms just in multiple messenger problems with multiple depots.

5. Conclusion

Results of computational experiments indicate higher efficiency of proposed insertion method than nearest neighbour algorithm described in [9]. The application of exchange algorithm significantly improved solutions obtained by nearest neighbour algorithm. Although in case of insertion method the improvement

References

[1] CORDEAU, J.-F., LAPORTE, G., ROPKE, S.: Recent Models and Algorithms for One-to-One Pickup and Delivery Problems, pp. 327-357. In: GOLDEN, B. L., RAGHAVAN, S., WASIL, E. A. (eds.): The Vehicle Routing Problem: Latest Advances and New Challenges. Springer, 2008.
[2] CORDEAU, J.-F., LAPORTE, G. A.: The Dial-a-Ride Problem: Models and Algorithms. Annals of Operations Research, vol. 153, pp. 29-46, 2007.
[3] DE PAEPE, W. E., LENSTRA, J. K., SGALL, J., SITTERS, J. A., STOUGIE, L.: Computer-Aided Complexity Classification of Dial-a-Ride Problems, INFORMS J. on Computing, vol. 16, pp. 120-132, 2004.
[4] CORDEAU, J.-F.: A Branch-and-Cut Algorithm for the Dial-a-Ride Problem. Operations Research, vol. 54, pp. 573-586, 2006.
[5] TOTH, P., VIGO, D.: Heuristic Algorithms for the Handicapped Persons Transportation Problem. Transportation Science, vol. 31, pp. 60-71, 1997.
[6] JORGENSEN, R. M., LARSEN, J., BERGVINDSOETIR, K. B.: Solving the Dial-a-Ride Problem Using Genetic Algorithms. J. of the Operational Research Society, vol. 58, pp. 1321-1331, 2007.
[7] FABRY, J., KOBZAREVA, M.: Multiple Messenger Problem. Proc. of Mathematical Methods in Economics ‘12, Karvina, pp. 141-147, 2012.
[8] FABRY, J.: Dynamic Messenger Problem. Communications - Scientific Letters of the University of Zilina, No. 4, pp. 66-69, 2007.
[9] PRIBYLOVA, L.: Heuristic Methods for Capacitated Messenger Problems (in Czech), Diploma thesis, University of Economics: Prague, 2014.
[10] CICKOVA, Z., REIFF, M., SURMANOVA, K.: Split Delivery Vehicle Routing Problem with Time Windows. Mathematical Methods in Economics 2014, Olomouc, pp. 128-132, 2014.