
Production Systems and Information Engineering
Volume 8 (2019), pp. 89–100
https://doi.org/10.32968/psaie.2019.006

CLUSTERING-BASED OPTIMISATION OF MULTIPLE TRAVELING SALESMAN PROBLEM

Anita Agárdi
University of Miskolc, Hungary
Institute of Information Science
agardianita@iit.uni-miskolc.hu

László Kovács
University of Miskolc, Hungary
Institute of Information Science
kovacs@iit.uni-miskolc.hu

[Accepted December 2017]

Abstract. The TSP is the problem to find the shortest path in a graph visiting every nodes exactly once and returning to the start node. The TSP is shown to be an NP-hard problem. To provide an acceptable solution for real life problems, the TSP are usually solved with some heuristic optimization algorithms. The paper proposes a clustering-based .problem decomposition algorithm to form the global route with merging of best local routes. Based on our test results, The proposed method can improve the efficiency of the standard heuristic methods for the TSP and MTSP problems.

Keywords: Traveling Salesman Problem, Clustering, Heuristic optimization

1. Introduction

In base Traveling Salesman Problem (TSP), a sales agent should visit all cities exactly once, except the a depo city which is the source and terminal of the route. The cities are represented with the vertices of a directed graph. The edges are weighted and the goal is to find a route visiting all cities and having a minimal total weight. The problem to build an optimal route corresponds to find an optimal permutation of the vertices creating a minimal length route. The TSP algorithm can be applied to solve many different problems, like

- planning of an optimal transportation route for both delivery and collection tasks [2],

- heuristic solving of the knapsack optimization problem [3],
- planning of optimal production sequence [1],
- printing press scheduling problem [4].

Based on the literature, the following variants of the TSP problem can be highlighted:

- base Traveling Salesman Problem (TSP): single depot position and single salesman,
- Multiple Traveling Salesman Problem (MTSP): single depot position and many salesmen,
- Multi Depot Multiple Traveling Salesman Problem (MDMTSP): many depots and many salesmen; every salesman is assigned to a single given depot.

The TSP is a NP-hard problem [10], that means it is at least as hard as the hardest problems in NP. The cost of a brute-force algorithm is in $O(N!)$, thus some heuristic should be used to find an approximation of the global optimum.

From the family of popular evolutionary algorithms, the following methods are used most widely to find the god approximation of the optimal route:

- Genetic algorithm [5]: the set of parameter vectors is optimized using selection, crossover and mutation operators.
- Particle swarm optimization [6]: more agents are generated which move randomly but the optimum found by them is reinforced by other members of the colony.
- Ant Colony Optimization [7]: more agents are generated which collaborate one another and with their environment.
- Tabu search [8]: the local search phase is extended with prohibitions (tabu) rules to avoid unnecessary position tests.

Considering the standard heuristic algorithms, we can highlight the following methods:

- Nearest neighbor algorithm [9]: it select the nearest unvisited node as the next station of the route.
- Pairwise exchange of edges [12]: two disjoint edges are removed from the route and two new edges are involved into the route to reduce the total cost.

In many real applications the nodes corresponds to points in the space, where the weights correspond to the distances between these positions. In this paper, we are focusing on the problem where the nodes represent points in the Euclidean space. We assume that there exists always a connection between any

node pair within the graph. Considering the TSP with nodes in Euclidean-space, we can observe that the optimal route usually connects neighboring nodes, i.e. the adjacent nodes are near to each others.

In this paper, we introduce a two-phase route planning method with clustering of the cities. In the first phase, the nodes are partitioned into disjoint clusters. For every cluster, a local optimum route is generated. In the second phase, these local routes are merged into a global route. The global route is then refined using some standard heuristic methods.

The different cities are clustered into disjoint groups where a group contains similar nodes. The similarity of two nodes is measured with the distance value between the corresponding positions in the Euclidean space. To merge the local routes into a global route, the optimum route of the clusters and the optimum connection edges must be determined within a separate optimization module. In Multiple Traveling Salesman Problem, we can use the clustering method to determine the set of nodes assigned to the same salesman. Every salesman must visit the cities of the same cluster.

The efficiency of the proposed two-phase TSP method is tested with the following heuristic TSP optimization methods: Hill-climbing algorithm; Nearest neighbor algorithm and Nearest Insertion algorithm.

2. Formal description of the TSP

The formal model of TSP can be given with the following linear programming description:

- N: number of nodes in the graph.
- u_i : the position of the i-th city in the solution path.
- D: distance matrix; d_{ij} is the weight of the edge from the i-th node to the j-th node; the distance values are non-negative values.
- X: adjacency matrix of the Hamilton cycle; $x_{ij} = 1$ if there is a directed edge in the path from the i-th node to the j-th node; otherwise $x_{ij} = 0$.
- the objective function of the path optimization is: $\sum_{i=1}^{N} \sum_{j=1}^{N} d_{ij} x_{ij} \rightarrow min$.
- every node has only one incoming edge: $\forall j (i \neq j) : \sum_{i=1}^{N} x_{ij} = 1$.
- every node has only one outgoing edge: $\forall i (i \neq j) : \sum_{j=1}^{N} x_{ij} = 1$.
- there is only a single tour covering all cities: $u_i - u_j + N x_{ij} \leq N - 1$.

In the case of multi-salesman traveling (MTSP) problem more than one cycles should be generated (each salesman has a separate cycle) and each node is visited only by one salesman. For MTSP, the formal model should be extended with the following elements:

- M : number of salesmen.
- constraints on the depo node (start and stop): $\sum_{i=1}^{N} x_{i0} = M, \sum_{j=1}^{N} x_{0j} = M$.

Due to the high complexity of TSP, there exists no algorithm for global exact optimization with polynomial cost. For example, the Held-Karp algorithm solves the problem in $O(n^2 2^n)$ complexity. In order to provide an acceptable solution for real life problems, the TSP are usually solved with some heuristic optimization problem.

In the existing TSP algorithms, we use some heuristics to find an optimum route. In our investigation, we use the following methods: hill climbing algorithm; nearest neighbor algorithm and nearest insertion algorithm.

Having a context parameter set position p_0 , the hill climbing method evaluates some neighboring positions and selects the position with the best improvement factor. This selected position will the next context position. The algorithm terminates if no neighboring position with better fitness value is found. In order to avoid weak local optimum positions, some random relocation mechanism is added to the base algorithm. In this case, several initial positions are generated and processed. The 2-opt and 3-opt switch operations are the most widely used methods to generate the neighboring positions. The 2-opt method select two random edges from the context route and replace them with two new matching routes. For example, having a route $A \to a \to B \to b$ with two selected edges, a and b, the neighbor route is given with $A \to b \to B' \to a$, where B' denotes the inverse traversing of B. The 3-opt switch selects 3 edges and replaces them with 3 new edges to get a neighboring route.

In the case of Nearest Neighbor algorithm [13] uses a simple and sometimes efficient method to generate the optimum route. The algorithm starts with a context node selected randomly. In the next step, the method evaluates all free nodes and selects the node nearest to the context node. This node will be the next element of the route. The algorithm will then process this node as a new context node. This greedy algorithm terminates if all the nodes are merged into the route. The MTSP variant of the Nearest Neighbor method partitions the nodes into disjoint groups belonging to the different agents. The simplest way to perform a partitioning is to set a size threshold nl on the single routes. This constraint means that the greedy algorithm presented before terminates if the nodes in the route equals to nl. After completing a local route, the method starts to build up the local route for the next agent. Another specialty of this algorithm is that every local route starts with the depo node.

Also the Nearest Insertion method builds up the route on an incremental way. It starts with an node selected randomly. Having a context route, the algorithm evaluates all free nodes and calculates the minimal length increase related to inserting the selected node. Unlike the Nearest Neighbor method, the selected node can be inserted into any positions within the context route. The free node with the best minimal length increase is inserted into the route. In the case of MTSP, the method initially starts the generation of local route for ever agent.

3. Clustering methods

The clustering produces a partitioning of the elements where similar objects are assigned to the same group while unsimilar objects should be mapped to different groups. The input data can be given on two different ways. One option is to use attribute vectors to describe the objects where the similarity is calculated from the attribute values. Another option is to define the similarity matrix of the objects directly. The elements with high similarity should be assigned to the same cluster.

There are three main types of clustering methods:

- partitioning methods,
- hierarchical methods,
- density-based methods.

In the case of partitioning methods, in every iteration, a new partitioning is generated. Having an initial partitioning, the quality is improved by re-assignment of the items to different clusters. The re-assignment iteration is terminated if the improvement value is getting below a threshold. The K-means algorithm is a widely used partitioning-based clustering method.

In the hierarchical clustering methods, we take some extremum partitioning (for example every item is a separate cluster). In every iteration step, the current partitioning will be refined or it will be more coarse. The HAC [14] a known example of this algorithm group.

In the case of density-based clustering, a item-density value is calculated for every item position (or for the whole item domain). The clusters are defined as maximal connected dense areas. Regarding the quality, the following properties should be met by the selected partitioning algorithm [15]:

- scalability,
- supporting arbitrary cluster shape,
- detection of outlayer (noise) items,
- efficiency for high dimensional item domains.

The K-means clustering partitions the items into a fixed number of clusters. The number of the clusters is given as an input parameter set by the users. Initially, the cluster centers are set randomly. In every iteration step:

- the items are assigned to the nearest center,
- a new center point is calculated for every cluster,
- if the old and new centers are within a threshold distance, the iteration terminates.

The new cluster center is calculated with

$$x_c = \frac{1}{N_c} \sum_{i_c} x_{i_c}$$

The algorithm optimizes the sum of squared errors value:

$$\sum_{c} \sum_{i_c} d(x_c, x_{i_c})^2$$

There are two main approaches in the hierarchical clustering methods. The first approach uses an agglomerative, bottom-up construction concept, while the second approach is based on a divisive, top-down algorithm. In the case of agglomerative method, every item is a separate cluster initially. In the iterative loops, the two clusters having the largest similarity are merged in to a new single cluster. The HAC method is the most widely used hierarchical clustering method [14], where the clusters are merged on a greedy way. Two different termination conditions can be given: minimal number of the clusters or the minimal similarity for merging. The following methods are main methods to measure the similarity (or distance) of two cluster.

- single linkage method: the distance of two clusters is equal to the distance of the two nearest points

$$d(c_1, c_2) = min_{x \in c_1, y \in c_2} d(x, y)$$

- complete linkage method: the distance of two clusters is equal to the distance of the two farthest points:

$$d(c_1, c_2) = max_{x \in c_1, y \in c_2} d(x, y)$$

- centroid linkage method: the distance of two clusters is equal to the distance of the two center points:

$$d(c_1, c_2) = d(x_{c_1}, x_{c_2})$$

- average linkage method: the distance of two clusters is equal to the average distance related to any possible point pairs:

$$d(c_1, c_2) = \frac{1}{|c_1||c_2|} sum_{x \in c_1, y \in c_2} d(x, y)$$

4. Clustering extended heuristic methods

We propose the extension of the standard heuristics methods with a clustering phase in order to decompose the problem domain into a set of smaller domains. This decomposition can be considered as the application of the 'divide and conquer' approach in solving of the TSP optimization problems. To analyze the efficiency of clustering-based decomposition phase, we have developed the following algorithms.

Clustered hill-climbing for TSP:

1. Clustering of the nodes into disjoint groups (K: no specific constraint).
2. Calculate the cluster centers.
3. Hill-climbing TSP on the set of clusters to determine the cluster-level route.
4. Determining the input/output ports of the clusters.
5. Hill-climbing TSP for every cluster to determine the inner-level routes.
6. Merging the inner-level routes into a global route based on the cluster-level route.

Clustered hill-climbing for MTSP:

1. Clustering of the nodes into disjoint groups (K : number of the agents).
2. Extending every cluster with the common depot element.
3. (Clustered) Hill-climbing TSP for every cluster to determine the local routes for every agent.

Clustered hill-climbing for MDMTSP:

1. Clustering of the nodes into disjoint groups (K : number of the agents).
2. Extending every cluster with the nearest depot element taken also the capacity constraints into account.
3. (Clustered) Hill-climbing TSP for every cluster to determine the local routes for every agent.

Clustered nearest neighbor for TSP:

1. Clustering of the nodes into disjoint groups (K: no specific constraint).
2. Calculate the cluster centers.
3. Nearest neighbor TSP on the set of clusters to determine the cluster-level route.
4. Determining the input/output ports of the clusters.
5. Nearest neighbor TSP for every cluster to determine the inner-level routes.
6. Merging the inner-level routes into a global route based on the cluster-level route.

Clustered nearest neighbor for MTSP:

1. Clustering of the nodes into disjoint groups (K : number of the agents).
2. Extending every cluster with the common depot element.
3. (Clustered) nearest neighbor TSP for every cluster to determine the local routes for every agent.

Clustered nearest neighbor for MDMTSP:

1. Clustering of the nodes into disjoint groups (K : number of the agents).
2. Extending every cluster with the nearest depot element taken also the capacity constraints into account.
3. (Clustered) nearest neighbor TSP for every cluster to determine the local routes for every agent.

Clustered nearest insertion for TSP:

1. Clustering of the nodes into disjoint groups (K: no specific constraint).
2. Calculate the cluster centers.
3. Nearest insertion TSP on the set of clusters to determine the cluster-level route.
4. Determining the input/output ports of the clusters.
5. Nearest insertion TSP for every cluster to determine the inner-level routes.
6. Merging the inner-level routes into a global route based on the cluster-level route.

Clustered nearest insertion for MTSP:

1. Clustering of the nodes into disjoint groups (K : number of the agents).
2. Extending every cluster with the common depot element. (Clustered) nearest insertion TSP for every cluster to determine the local routes for every agent.

Clustered nearest insertion for MDMTSP:

1. Clustering of the nodes into disjoint groups (K : number of the agents).
2. Extending every cluster with the nearest depot element taken also the capacity constraints into account.
3. (Clustered) nearest insertion TSP for every cluster to determine the local routes for every agent.

In the case of multi-depot TSP variants, the depots may be fixed in advance or it can be located to any nodes. In our algorithm, we used the cluster-center depo location mechanism.

The implemented test framework provides the following functions:

- generation of test data,
- test data file level input/output,
- execution of the mentioned standard heuristic methods,
- solving TSP, MTSP and MDMTSP problems,
- adding cluster-based optimization,
- representation of the results in graph formats.

5. Test results

The evaluation tests were executed for small and medium sized problem domains with the following parameters:

Table 1. Test parameters

	N: number of nodes	A: number of agents	D: number of depots
small domain	10–100	1–10	1–10
medium domain	100–1000	1–100	1–100

In the tests, two main quantities were analyzed:

- length of the generated route (L),
- execution time (T) in ms.

The results for (N:1000, A:10, D:1) are summarized in the Table 2. The column CL denotes whether the algorithm includes a clustering phase or not.

Table 3 shows the summarized results for medium size input domain (N:1000, A:100, D:1).

The analysis of the test results show, that

- the clustering provides significantly improvement in fitness efficiency for the complex MTSP and MDMtSP problems,
- the clustering can improve the fitness efficiency of the standard heuristic methods for the TSP problem, too,
- the fitness efficiency of the proposed low cost clustering optimization module is comparable with the fitness efficiency of the complex compound optimization algorithms requiring a higher execution cost,
- the clustering phase requires additional 20%–80% computation costs.

Based on the performed experiences, the proposed cluster optimization phase is a promising approach to enhance the efficiency of the complex TSP problems. In the next phase of our investigation, we will focus on the development of an improved TSP-adjusted clustering method.

Table 2. Test parameters for small number of agents

method	CL	T tsp	T mtsp	T mdmtsp	L tsp	L mtsp	L mdmtsp
nearest neighbor	N	125	83	125	2,787	3,603	3,234
nearest neighbor	Y	105	125	152	3,151	2,978	3,053
nearest insertion	N	31	43	105	3,062	4,294	3,795
nearest insertion	Y	36	56	123	3,255	3,784	3,068
hill climbing	N	16,124	24,014	24,012	3,538	10,051	10,421
hill climbing	Y	18,154	17,297	18,354	2,626	3,099	2,673
nearest neighbor + hill climbing	N	16,154	25,341	25,887	2,530	3,070	2,896
nearest neighbor + hill climbing	Y	17,895	15,781	15,443	2,611	2,978	2,605

6. Conclusion

For TSP problems with high number of nodes, the multi layered optimization is superior to the single level optimization. In the proposed multi layered optimiza tion, the node set is partitioned into clusters or into hierarchy of clusters. For each cluster, a separate local TSP optimization is performed and then the local routes are merged into a global route. Based on the test experiments the proposed method is superior to the single level optimization method for both the TSP and MTSP problems

Acknowledgments. The described article/presentation/study was carried out with the support of the EFOP-3.6.1-16-00011 Younger and Renewing University InnovativeKnowledge City institutional development of the University of Miskolc aiming at intelligent specialisation project implemented in the framework of the Szechenyi 2020 program. The realization of this project is supported by the European Union, co-financed by the European Social Fund.

Table 3. Test parameters for medium number of agents

method	CL	T tsp	T mtsp	T mdmtsp	L tsp	L mtsp	L mdmtsp
nearest neighbor	N	125	62	62	2,784	10,122	3,889
nearest neighbor	Y	17	12	93	2,612	9,620	2,976
nearest insertion	N	31	31	26	3,062	13,425	7,182
nearest insertion	Y	181	125	62	3,405	9,994	2,784
hill climbing	N	16,124	24,452	24,002	3,538	22,974	14,762
hill climbing	Y	30,045	33,012	31,034	2,643	9,355	2,597
nearest neighbor + hill climbing	N	16,164	24,514	24,881	2,530	8,905	3,081
nearest neighbor + hill climbing	Y	33,012	32,012	30,875	2,625	9,322	2,602

REFERENCES

[1] Jeong, EY., Oh, S.C., Yeo, YK. et al. Application of traveling salesman problem (TSP) for decision of optimal production sequence *Korean J. Chem. Eng.* ,(1997) Vol 14, pp 416 -421. *https://doi.org/10.1007/bf02707062*

[2] Berbeglia, G., Cordeau, J. F., Gribkovskaia, I., Laporte, G. Static pickup and delivery problems: a classification scheme and survey. *Top*, Vol 15(1), (2007), pp. 1-31. *https://doi.org/10.1007/s11750-007-0009-0*

[3] Laporte, G., Martello, S. The selective travelling salesman problem. *Discrete applied mathematics*, Vol 26(2-3), (1990), pp. 193-207. *https://doi.org/10.1016/0166-218x(90)90100-q*

[4] Gorenstein, S. Printing press scheduling for multi-edition periodicals. *Management Science*, Vol 16(6) (1970), pp. B-373. *https://doi.org/10.1287/mnsc.16.6.b373*

[5] Grefenstette, J., Gopal, R., Rosmaita, B., Van Gucht, D. Genetic algorithms for the traveling salesman problem. *In Proceedings of the first International Conference on Genetic Algorithms and their Applications* (1985), pp. 160-168.

[6] Wang, K. P., Huang, L., Zhou, C. G., Pang, W. Particle swarm optimization for traveling salesman problem. *In Machine Learning and Cybernetics*, (2003) Vol. 3, pp. 1583-1585. *https://doi.org/10.1109/icmlc.2003.1259748*

[7] Dorigo, M., Gambardella, L. M. Ant colonies for the travelling salesman problem. *biosystems*, Vol 43(2), (1997), pp. 73-81. *https://doi.org/10.1016/s0303-2647(97)01708-5*

[8] Malek, M., Guruswamy, M., Pandya, M., Owens, H. Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem, *Annals of Operations Research*, Vol 21(1), (1989) pp. 59-84.*https://doi.org/10.1007/bf02022093*

[9] Lin, S., Kernighan, B. W. An effective heuristic algorithm for the traveling-salesman problem, *Operations research*, Vol 21(2), (1973) pp. 498-516 *https://doi.org/10.1287/opre.21.2.498*

[10] Held, M.; Karp, R. M., A Dynamic Programming Approach to Sequencing Problems, *Journal of the Society for Industrial and Applied Mathematics* Vol. 10 (1962), 196–210. *https://doi.org/10.1145/800029.808532*

[11] Sumanta Basu, Swarm Optimization Algorithm for the Traveling Salesman Problem, *EvoCOP 2006: Evolutionary Computation in Combinatorial Optimization* (2006), 99–110.

[12] C. Nilsson., Heuristics for the traveling salesman problem., *Tech. Report, Linköping University, Sweden* (2003),

[13] Arthur E. Carter , Cliff T. Ragsdale, A new approach to solving the multiple traveling salesperson problem using genetic algorithms, *European Journal of Operational Research 175* (2006), 246–257. *https://doi.org/10.1016/j.ejor.2005.04.027*

[14] Maria-Florina,B, Liang , Y., Gupta,P. , Robust Hierarchical Clustering, *Journal of Machine Learning Research* , Vol 15, (2014), pp. 4011-4051.

[15] S. Saitta, B. Raphael, I.F.C. Smith, A Bounded Index for Cluster Validity, *Machine Learning and Data Mining in Pattern Recognition*, Vol 4571 (2007), pp. 174-187.