Solving Min-Max MTSP using Evolutive Algorithms and Clustering

Anda Buinoschi MOC2, Bogdan Luncaşu MOC1

Abstract

This paper showcases a method for solving Min-Max Multiple Traveling Salesman making use of Ant Colony Optimization, Simulated Annealing and K-means clustering algorithm. In Min Max MTSP, it is needed to minimize the maximum tours for each salesman, balancing the cost among them. This method proves comparable results to one of the available benchmarks on some instances. In the upcoming sections we introduce the problem, describe the methods, present the experiments' results and compare them with the benchmark.

1 Introduction

Traveling Salesman Problem is one of the most studied problem in combinatorial optimization, due to the complexity of finding good solutions for a large amount of cities in a timely manner. It states that: a traveling salesman has to visit a number of given cities in the shortest tour distance possible, visiting each city exactly once. The general case is MTSP in which you have more salesmen and it is needed to find subtours for them serving a set of cities. Being in the class of NP hard problems, MTSP solution time grows exponentially with the increase in input data points. As a result, the classical optimization procedures are not adequate for this problem especially for large dimensions. In other words, as the size of the problem increases, the complexity of the problem also increases rapidly as well. We present a method for this problem by combining clustering methods with combinatorial optimization algorithms.

2 Methods

2.1 Algorithms

2.1.1 K Means

K means algorithm is a method used for clustering a set of points with a label according to a similarity measure. Usually, if the data points are in an Euclidian space, then it is used their distance as similarity measure.
Given a set of data points $D = \{x_1, x_2, ..., x_n\}$, the algorithm iteratively search

for an optimal centroid for k clusters, the latter being given as parameter. The k centroids are chosen initially randomly from the search space or from the data points given. Assigning each point to their clusters according to the closest centroid, the latter are adjusted by recomputing their position, converging to an optimal solution in the end.

2.1.2 Simulated annealing

Simulated Annealing algorithm is a popular metaheuristic algorithm used for solving discrete and continuous optimization problems. The key feature lies in finding the right parameters for temperature, the cooling system and α which is used in the cooling function. Its origins are in statistical mechanics and its fundamental idea is to accept moves resulting in worse quality solutions than the current one to escape from local optima. The probability of accepting such a move is decreased through the temperature parameter. The Metropolis acceptance criterion, which models how a thermodynamic system moves from one state to other in which the energy is minimized, is used for accepting a worse neighbourhood or not. In order to apply the algorithm to a specific problem, it is needed to define a neighbourhood structure and the cooling schedule.
The cooling schedule defines the way in which the temperature is going to be decreased and it is crucial for the success of the search. A very low cooling schedule will take too many iterations to reach the global optima and if the number of iterations is limited, then this can lead to an unsuccessful result. On the other hand, a too fast cooling schedule can get the algorithm trapped in a local optima. The most common cooling schedule is geometric cooling schedule, which can be described by the formula $T_{k+1} = \alpha \cdot T_k$, where T is the temperature value, k is considered the iteration and *alpha* is the cooling rate. In this type of function, *alpha* has to be smaller but close to 1, otherwise this would lead to an accelerated cooling system. Usually, for this function *alpha* is choosing from the interval $[0.8, 0.99]$. Another cooling schedule is the logarithmic schedule $T_{k+1} = \frac{\alpha T_k}{ln(1+k)}$. Its convergence to reach the global optima is proved for $\alpha = 1$ but it is such a slow cooling schedule that is rarely used in practice. Even though smaller values for *alpha* can speed it up, it is considered to be slow in general. The pseudocode for the algorithm can be found below:

Algorithm 1: Simulated Annealing algorithm

```
t = 0;
initialize temperature T;
select a current candidate solution (bitstring) v_c at random;
evaluate v_c;
repeat
    repeat
        select at random v_n - a neighbour of v_c;
        if eval(v_n) is better than eval(v_c) then
            v_c = v_n;
        else if random([0,1)) < exp(-|eval(v_n)-eval(v_c)|/T) then
            v_c = v_n;
    until (termination-condition);
    T = g(T, t);
    t = t + 1;
until (halting-criterion);
```

For our paper we used geometric cooling schedule and for the neighbourhood structure we chose 2opt method, used for discrete values.

2.1.3 Ant Colony Optimization

The Ant Colony Optimization is very popular among the combinatorial problems on graph structures because it was inspired by some experiments on ant species which were trying to find the shortest path from their nest to the place where they could find food through a process called stigmergy. This process defines the environmental changes through the pheromone they were leaving on their way in such manner that the other ants from the colony were attracted to go on the same path since this is their communication method. The pseudocode in a short brief is the following:

Algorithm 2: Ant Colony Optimization algorithm (short presentation)

```
set parameters, initialize pheromone trails;
while termination condition not met do
    ConstructAntsSolutions;
    ApplyLocalSearch;
    UpdatePheromones;
end
```

The main phases of the algorithm constitute the ants' solution construction and the pheromone update. Each ant k from the colony has the following properties: it exploits the construction graph in the search for optimal solutions, it has a memory about the path it followed, a starting state and an ending state, selects a move by applying a probabilistic decision rule and updates the pheromone trail associated with the used components. Each ant acts concur-

rently and independently and that although each ant is complex enough to find a probably poor solution, the good solutions can only emerge as the result of the collective interaction within the colony.
In the algorithm, each ant concurrently builds a tour. They start from a node and for the construction step, each of them applies a probabilistic action choice rule to decide which node to visit next. In particular, the probability to visit node j from node i is the following:

$$p_{ij} = \frac{[\ \tau_{ij}]^{\ \alpha} \cdot [\ \eta_{ij}]^{\ \beta}}{\sum_{l \in N_i} [\ \tau_{il}]^{\ \alpha} \cdot [\ \eta_{il}]^{\ \beta}},$$

if $j \in N_i$, where $\eta_{ij} = \frac{1}{d_{ij}}$ is a heuristic value that is available a priori, α and β are parameters which determine the relative influence of the pheromone trail and the heuristic information, and N_i represent the feasible neighbourhood of the ant when being in node i; the probability of choosing a node outside N_i is set to 0. If $\alpha = 0$ then the closest cities are more likely to be visited, corresponding to a stochastic greedy algorithm. If $\beta = 0$, only pheromone amplification will be at work. This generally leads to poor results and for values $\alpha > 1$ it leads to rapid emergence and stagnation, a situation in which all the ants are following the same path and construct the same tour. Usually, α is set to 1 and β ranges from 2 to 5.
After the ants constructed their tour, the pheromone trails are updated. This step consists in pheromone evaporation (in order to avoid the accumulation of the pheromone trails and "forget" bad decisions previously taken in the exploration step) and then add the pheromone on the arcs the ants have crossed in their tours. Pheromone evaporation is define by

$$\tau_{ij} = (1 - \rho)\tau_{ij}, \forall (i, j) \in L,$$

where L is the set of arcs that connect the nodes of the graph in the problem and $0 < \rho \leq 1$ is the pheromone evaporation rate. After evaporation step, all ants deposit pheromone on the arcs they have crossed in their tour:

$$\tau_{ij} = \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \forall (i, j) \in L,$$

where $\Delta\tau_{ij}^{k}$ is the amount of pheromone ant k deposits on the arcs it has visited. It is defined by:

$$\Delta\tau_{ij}^{k} = \begin{cases} 1/C^k & \text{if arc } (i,j) \text{ belongs to } T^k \\ 0 & otherwise \end{cases},$$

where C^k is the length of the tour T^k built by the k-th ant; it is computed as the sum of the lengths of the arcs belonging to T^k. This means that the better an ant's tour is, the more pheromone the arcs belonging to this tour receive. In general, arcs that are used by many ants and which are part of short tours, receive more pheromone and therefore more likely to be chosen by ants in future iterations of the algorithm.

2.2 Our method

Our method consisted in running KMeans clustering algorithm, with the parameter k set to the number of travel salesmen that our problem has, then running Simulated Annealing and Ant Colony Optimization on these resulted clusters. The idea of creating these clusters was to find groups of cities and nodes from which each of the travel salesmen start creating their tours. We excluded the first point of the dataset (considered to be depot) while creating the clusters, and then constructed the tours starting with the depot point. On short, each cluster plus the depot point represented instances of classical TSP problem. These clusters of cities have the benefit of being conglomerate and they can lead to good results (short tour costs within clusters), almost comparable to the benchmark. The disadvantage is that some centroids of the clusters are far apart from the depot point and this reflects in our results.

3 Experiments

The experiments were run on the following datasets: *eil51*, *berlin52*, *eil76*, *rat99* which can be accessible on TSPLib.

3.1 Simulated Annealing with clustering

For the simulated annealing the following parameters were chosen:

- temperature T = 1200
- cooling rate $\alpha = 0.9995$
- geometric cooling schedule

3.2 Ant Colony Optimization with clustering

For the Ant Colony Optimization the following parameters were chosen:

- pheromone accepting rate $\alpha = 0.8$
- heuristic accepting rate $\beta = 2.5$
- pheromone evaporation rate $\rho = 0.5$
- number of ants in the colony: 100
- number of generations: 100

3.3 Benchmark

Our benchmark consists in the results run on CPLEX using other type of optimization method, whom authors are: Mihaela Breabăn, Mădălina Răschip and Raluca Necula. These results are shown in the table 3.

Instance Name	n (# cities)	m (# salesmen)	Optimal MinMax	Optimal Cost
eil51	51	2	[204.199, 267.417]	471.615
		3	[145.299, 207.052]	541.351
		5	[90.584, 164.939]	597.936
		7	[52.368, 137.929]	712.048
berlin52	52	2	[3504.561, 5792.916]	9297.478
		3	[2621.278, 3504.561]	9549.496
		5	[1353.062, 3482.195]	11233.968
		7	1067.030, 2879.221]	12868.747
eil76	76	2	[283.372, 367.943]	651.315
		3	[213.911, 297.207]	755.314
		5	[155.375, 186.110]	872.618
		7	[94.907, 193.164]	996.813
rat99	99	2	[860.188, 877.018]	1737.205
		3	[429.195, 888.594]	1980.749
		5	[373.163, 671.721]	2563.429
		7	[211.729, 598.262]	2870.676

Table 1: Value results after running KMeans with Simulated Annealing

Instance Name	n (# cities)	m (# salesmen)	Optimal MinMax	Optimal Cost
eil51	51	2	[262.191, 285.781]	547.973
		3	[177.747, 218.622]	599.363
		5	[81.578, 192.115]	650.346
		7	[72.546, 164.789]	810.192
berlin52	52	2	[4096.385, 5766.348]	9862.734
		3	[2504.381, 4046.373]	10048.535
		5	[1245.062, 4444.853]	11683.358
		7	[1200.664, 3571.886]	14227.138
eil76	76	2	[286.730, 369.189]	655.919
		3	[191.295, 312.171]	721.069
		5	[130.945, 221.989]	885.095
		7	[97.864, 201.694]	1056.032
rat99	99	2	[831.072, 1009.782]	1840.855
		3	[440.119, 914.026]	2050.835
		5	[381.065, 743.852]	2826.066
		7	[222.059, 707.777]	3556.046

Table 2: Value results after running KMeans with Ant Colony Optimization

Instance Name	n (# cities)	m (# salesmen)	Optimal MinMax	Optimal Cost
eil51	51	2	222.73	444.33
		3	[150.70, 159.57]	477.15
		5	[96.91, 123.96]	615.19
		7	[75.91, 112.07]	762.83
berlin52	52	2	[4049.05, 4110.21]	8217.94
		3	[2753.63, 3244.37]	9591.15
		5	[1671.69, 2441.39]	12084.90
		7	[1272.06, 2440.92]	16768.79
eil76	76	2	280.85	561.48
		3	[186.34, 197.34]	587.65
		5	[117.61, 150.30]	748.43
		7	[88.35, 139.62]	964.69
rat99	99	2	[620.99, 728.71]	1456.95
		3	[426.25, 597.17]	1780.48
		2	[271.91, 469.25]	2336.22
		3	[210.41, 443.91]	307 4.3

Table 3: Our benchmark, see References[3]

4 Conclusion

In conclusion, our method which combines the simple algorithm of KMeans, Simulated Annealing and Ant Colony Optimization doesn't lead to very good results but it is comparable with the benchmark for some of the instances. From our results' tables, it can be observed that the Min Max problem is not fully solved, since we didn't combine the algorithms with an inter-TSPs tours heuristic. There is room for improvement and some ideas are: exchange nodes between clusters, removing a node from a TSP instance which is overwhelmed by the cost tour and give it to a shorter-tour TSP instance and others.
This method doesn't lead to the best solutions but it may lead to a good approximation of optima for some instances.

References

[1] TSPLib for data points
http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/

[2] 2opt method for generating neighbours in Simulated Annealing
https://en.wikipedia.org/wiki/2-opt

[3] Breaban Mihaela, Raluca Necula, Madalina Raschip for benchmark results
https://profs.info.uaic.ro/~mtsplib/MinMaxMTSP/index.html

[4] Marco Dorigo, Thomas Stützle, "Ant Colony Optimization", 2004
https://web2.qatar.cmu.edu/~gdicaro/15382/additional/aco-book.pdf

[5] Breaban Mihaela, Nature Inspired Methods class course, 2020
https://profs.info.uaic.ro/~pmihaela/MOC/

[6] MIT Open Course Ware, "Multidisciplinary System Design Optimization", Olivier de Weck, 2010
https://ocw.mit.edu/courses/institute-for-data-systems-and-society/ids-338j-multidisciplinary-system-design-optimization-spring-2010/lecture-notes/MITESD_77S10_lec10.pdf

[7] Wikipedia site for Simulated Annealing and Ant Colony Optimization
https://en.wikipedia.org/wiki/Simulated_annealing
https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms