A TEAM ANT COLONY OPTIMIZATION ALGORITHM FOR THE MULTIPLE TRAVELLING SALESMEN PROBLEM WITH MINMAX OBJECTIVE

Ilari Vallivaara
Intelligent Systems Group (ISG),
University of Oulu, Finland

ABSTRACT

In this paper, a Team ant colony optimization algorithm (TACO) is proposed for the multiple travelling salesman problem with MinMax objective. The novel idea is to replace every ant in an ant colony optimization algorithm, for example Ant Colony System [1], with a team of ants and letting those teams construct solutions to the multiple travelling salesman problem. The simulation results show that the proposed algorithm outperforms existing neural network based approaches in solution quality. Furthermore, the presented experiments demonstrate the feasibility of the proposed approach in multi-robot path planning.

1 Introduction

1.1 TSP and MTSP

The travelling salesman problem (TSP) [2] is a famous and computationally very hard optimization problem, in which one must find the shortest route through n cities visiting each city exactly once and returning to the starting city. In graph theoretic terms the problem can be stated as finding a Hamiltonian cycle of the least weight in a complete weighted graph, where the nodes represent the cities, the edges represent the roads, and the weights represent the distance (or cost travelling) between cities.

Because enumerating all possible tours and other exact methods usually become computationally impractical (or even impossible) very quickly when the number of cities grows, algorithms giving near-optimal solutions are often acceptable. Basically the methods can be divided into tour construction and tour improvement heuristics. A good overview of exact and approximate algorithms used can be found in [3].

The multiple travelling salesman problem (MTSP), where m salesmen visit the n cities (each city is visited only once), is a generalization of the TSP ($m = 1$ in the TSP). Solutions to the MTSP consist of tours, called routes, on subsets of the original city set. Usually the salesmen start from the same city, but one can easily define different starting cities for each one of them. In this paper, the MinMax objective is used as the criterion to evaluate the solutions to the MTSP, *i.e.*, the length of the longest route traveled by a salesman.

The vehicle routing problem (VRP) [4] is closely related to the MTSP. The difference is that in the VRP the solutions are usually constrained, *e.g.* each salesman (vehicle) has a limited capacity and each city has a demand. Therefore the number of salesmen used becomes one of the factors to be optimized, whereas in the MTSP the number of salesmen is fixed. The MTSP can be seen as a special case of the VRP with no constraints, *e.g.* infinite capacities or zero demand.

1.2 MTSP with MinMax Objective

In many real life cases the objective is to complete the given mission in minimum time using all resources available, *i.e.* making the workload distribution as even as possible. In rescue operations [5] this means searching the area in minimum time, and in cooperative robot cleaning tasks getting the area cleaned in minimum time, for example. Optimization problems of this type can usually be restated as a MTSP instance with MinMax objective, *i.e.* finding the routes that minimize the longest route length.

This strongly motivates using the MinMax criterion to evaluate the solutions, even if there often is a single route solution which minimizes the sum of the route lengths. This kind of MinMax approach is taken in [6], [7] and [5], for example. Some authors [8] prefer minimizing the sum of squares of route lengths in order to distribute the workload. However, in some cases minimizing the sum of squares results in uneven workload distribution compared to the MinMax criterion, which has been illustrated in Fig. 1.

1.3 Ant Colony Optimization

Ant colony optimization (ACO) takes its inspiration from real ants in nature and their foraging behaviour. Ants use chemical traces, *i.e.* pheromone, to coordinate their search for food. When an ant finds a food source and returns to the nest, it leaves (stronger) pheromone traces on the ground. Those traces guide other ants' essentially random movement by biasing them to choose paths with high concentration of pheromone. Eventually paths leading to good food sources are filled with pheromone and followed by most of

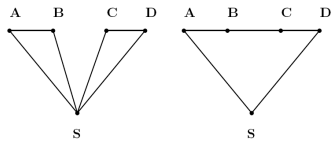

Figure 1. A simple example of a situation where the Min-Max criterion (left) is more suitable than the square sum criterion (right) in terms of workload distribution. Two salesmen have to visit cities A, B, C, and D, starting from and returning to city S.

the ants leaving the nest. This also allows ants to optimize path lengths because pheromone will accumulate faster on shorter paths leading to a food source.

1.3.1 Ant System

The original *Ant System* (AS), proposed by Dorigo et al. in [9], uses this idea to solve the TSP by letting a large number of artificial ants randomly construct tours by consecutively moving from node to node, and then letting them deposit pheromone on the used edges depending on their tour length: short tours receiving more pheromone. Edges shared by many short tours are therefore getting the most pheromone. The tour construction in the following iterations is then biased by the pheromone deposited on the edges. Each time an ant k, located in node i, has to choose the next node in its tour, AS uses the following random proportional rule to determine the probability p_{ij}^k of node j to be chosen as the next node:

$$p_{ij}^k = \begin{cases} \dfrac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum\limits_{u \in J_k} [\tau_{iu}]^\alpha \cdot [\eta_{iu}]^\beta}, & \text{if } j \in J_k; \\ \\ 0, & \text{otherwise;} \end{cases} \quad (1)$$

where τ_{ij} is the pheromone on edge (i, j), $\eta_{ij} = d(i, j)^{-1}$ is the inverse of the distance between nodes i and j. Parameters α and β are used to control the importance of the pheromone concentration compared to distance, and J_k is the set of feasible nodes.

Before the first iteration, a small amount of pheromone is deposited on each edge, and after each iteration, the pheromone on each edge is evaporated to avoid unlimited accumulation. For further details about the original algorithm, reading [9] or [10] is recommended.

Later the idea behind AS has been used to solve many kinds on optimization problems like the vehicle routing problem (VRP) [11] and the quadratic assignment problem (QAP) [12]. The extensions of AS, like *Ant Colony System* (ACS) [1] and *Max-Min Ant System* (MMAS) [13], greatly improve its performance. Overview of ACO algorithms can be found in [10].

1.3.2 Ant Colony System

Ant Colony System (ACS) is an extension of AS proposed by Dorigo and Gambardella in [1]. There are few main differences to AS. ACS uses global pheromone update only for the best tour found so far (T^{bs}). ACS also utilizes pseudorandom proportional rule (2) that results in aggressive search in the neighbourhood of T^{bs}, and local pheromone update. ACS typically uses fewer ants ($N = 10$) compared to AS ($N = n$).

In ACS, when ant k, located in node i, has to choose its next node j, it uses the following pseudorandom proportional rule given in Eq. (2):

$$j = \begin{cases} \text{argmax}_{l \in J_k} \left\{ \tau_{il} [\eta_{il}]^\beta \right\}, & \text{if } q \leq q_0; \\ \\ J, & \text{otherwise;} \end{cases} \quad (2)$$

where q is a random number drawn from $[0, 1]$, $q_0 \in [0, 1]$ is a parameter, and J is a random variable given by (1) (with $\alpha = 1$). In other words, with probability q_0 the ant chooses the best node according to the pheromone and the distance, and with probability $(1 - q_0)$ it chooses its next node randomly using equation (1). In ACS only one ant, the best-so-far ant, is allowed to deposit and evaporate pheromone after each iteration. This global update is implemented by using Eq. (3):

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho \Delta \tau_{ij}^{bs}, \qquad \forall (i, j) \in T^{bs}, \quad (3)$$

where $\Delta \tau_{ij} = 1/C^{bs}$ is the inverse of the tour length of the best-so-far ant. In addition to the global update rule the ants use the following local pheromone update rule that is used immediately after an ant crosses an edge (i, j) during the tour construction:

$$\tau_{ij} \leftarrow (1 - \xi)\tau_{ij} + \xi \tau_0, \quad (4)$$

where ξ and τ_0 are control parameters. Values $\xi = 0.1$ and $\tau_0 = 1/nC^{nn}$, were found empirically. The value $1/nC^{nn}$ is also the initial value for the pheromone trails, where C^{nn} is the length of the tour constructed by the nearest neighbour heuristic. The local update makes the used edges less desirable and increases the exploration of edges not visited yet.

ACS was chosen to be the base of the proposed algorithm due to its aggressive nature. Other extensions, mainly MMAS and its variants, could also be used.

2 The Proposed Algorithm for The MTSP

The proposed Team ant colony optimization algorithm (TACO) is basically a generalization of ACS for the MTSP. The idea is to replace every ant with a team of ants, consisting of m members, making the total number of ants equal to Nm. Teams are then used to construct solutions in parallel. Each team has its own visited list and its members construct routes by moving in turns to nodes not yet visited

by the team. The moving member is chosen according to the route lengths, most of the time being the one with the minimum route length.

The one difference between ACS and TACO with $m = 1$ is that in ACS the ants can be placed initially randomly. Because this is not possible in TACO and in MTSP, it is natural to place the ants initially in the starting cities. If the salesmen (or some of them) are using the same starting city, this can be achieved by making a copy of the city for each extra salesman.

In the VRP, the constraints can be used to determine when to end one route and start constructing the next one [11], [10], [14]. This is not possible in the MTSP, because the capacities of the salesmen are not limited. The team approach does not need the information about the route lengths, as the routes are constructed simultaneously by team members.

Like most ACO algorithms, TACO can be easily converted to solve many kinds of problems. Because of its general nature, it can be easily implemented on instances with more than one starting or ending city, instances where the routes aren't closed, and instances that are asymmetric or non-Euclidean.

2.1 TACO and Route Construction

During the initialization phase the pheromone level is set to $\tau_0 = 1/nC^{nn}$ on each edge as suggested in ACS. Before beginning the tour construction, all cities are set non-visited for each team. Then each member k of a team is placed in the corresponding starting city s_k, and all starting cities are marked as visited for that team.

During the route contruction the route length travelled by each member k is stored in $RL(k)$. The main idea is to move always the member with the minimum route length to guarantee approximately even route lengths. Unfortunately, in many cases this results in clearly non-optimal solutions as can be seen from Fig. 2. Therefore an extra check is needed before letting an ant move.

After ant k, located in x_k, with the minimum route length has chosen its destination city j according to the rules (2) and (1), it is checked if any other member l, located in x_l, from the team could add the chosen city to its route and end up with better total route length. In other words, if for any ant l

$$d(x_l, j) + d(j, s_l) + RL(l) < d(x_k, j) + d(j, s_k) + RL(k) \quad (5)$$

holds, that ant is allowed to make its move first. It then chooses the next city for itself (not necessarily j) and moves there. If multiple members satisfy (5), then the one with minimum $d(x_l, j) + d(j, s_l) + RL(l)$ is chosen. The check above prevents the algorithm from forcing non-optimal solutions. Other rules for this could also be used especially in non-Euclidean cases.

As suggested in ACS, the use of precalculated *nearest-neighbour* list for each city, improves the perfor-

Algorithm 1 TACO algorithm

```
% Initialization phase

% Route construction
reset all teams
for all teams (simultaneously) do
   for move = 1 to (n - number of starting cities) do
      choose moving member
      choose destination  % (1) & (2)
      check for better  % (5)
      move ant
      local pheromone update  % (4)
   end for
end for
move ants to their starting cities (and local update)

% Local optimization phase
2-opt routes for each team
evaluate
3-opt the routes of the best team of iteration

% Global update
evaluate
global pheromone update  % (3)

if (end conditions = true) then
   print best solution found
else
   goto Route construction
end if
```

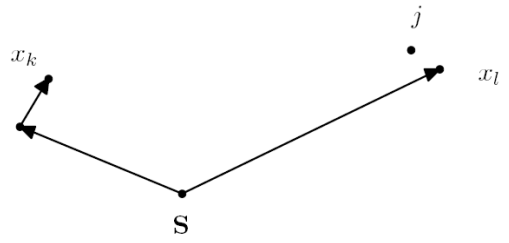

Figure 2. If another member l can add city j to its route and return to starting city ending up with better total route length, then it is allowed to make its move first, see Eq. (5). If ant k, having the shorter route length, was forced to make its move first, it would clearly result in non-optimal solution.

mance of the algorithm in both running times and solution quality. This list is of constant length cl and is called *candidate list*. Instead of choosing their next city from all the cities, ants choose an unvisited city only from the cl closest cities, making the city selection much faster. The set of feasible cities J_k is therefore the intersection of unvisited cities and cities belonging to the candidate list of city k. Only after all cities from J_k are visited, ants choose outside the list. In that case, the closest city is automatically chosen.

After an ant finally moves from city i to j, the local update rule (4) is applied on edge (i, j), ant's location is updated, j is marked as visited, and edge (i, j) is added to the route under construction.

The m ants from a team then continue moving in turns constructing m routes. This is repeated until all cities are visited. Then each ant moves to its starting city.

All of the above is applied to every team independently. There is a choice to let the teams construct the routes in parallel or sequentially. At the time of [10] there was no experimental evidence in favor of one choice or another. As in most ACS implementations, parallel construction is used in TACO.

2.2 Local Search

Combining local search with ACO algorithms usually improves their performance significantly. TACO is not an exception. After the teams have constructed their routes, the routes are taken to their local optimum by applying so called 2-opt procedure. It basically removes two edges from a route and then checks if connecting the route in the other possible way makes the solution better. This is implemented on every pair of edges. A standard speedup technique, the use of nearest-neighbor list of constant length (2-opt depth), is used to limit the edge removing to only edges near each other and to keep the complexity subquadratic.

After brought to their local optimum, the solutions are evaluated by MinMax criterion. The best solution is then allowed to use 3-*opt* to find improvements that can be achieved by removing three edges (in 3-opt depth) and then connecting the route in all possible ways. In both local optimizations the first improvement found is accepted.

This combination of two local search procedures tries to balance between speed and quality. Other combinations are also possible. For example, in [1] ACS is coupled with 3-opt which is applied to every solution, giving very good results. Even the sophisticated Lin-Kernighan [15] could be used in combination with the proposed algorithm. For details on implementing the local optimization and possible speedup techniques, see [16].

Other local optimization procedures suitable for the MTSP could be, for example, node-exchanges and node-moves between different routes. However, in this paper and the computational results presented here, only 2-opt and 3-opt are used in order to test the main idea of the algorithm.

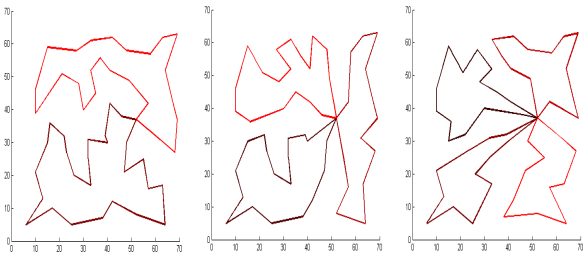

Figure 3. Solutions to TSPLIB instance Eil51 for 2, 3, and 4 salesmen, generated by TACO

2.3 Global Update and Evaluating Solutions

Although minimum square sum is not the criterion the solutions are evaluated by, test runs suggest that approving improvements by both MinMax and square sum criterions helps the algorithm to escape local optima, especially on early stages of the run. Apparently good square sum solutions lead to good MinMax solutions and vice versa. It should also be noted that most of the time an improvement is an improvement by both criterions. In the computational experiments presented in this paper, solutions are evaluated by both criterions, and if an improvement is found, it becomes the best-so-far team and is allowed to deposit pheromone.

The above suggests that if in the future TACO is implemented on some rank-based ACO algorithm, ranks can be based on both criterions.

After evaluating solutions from the current iteration, the global pheromone update (3) is applied, adding pheromone on all edges belonging to T^{bs}, which is the solution of the best-so-far team. A natural choice for the amount of pheromone deposited is $\rho\Delta\tau_{ij}^{bs} = \rho\ 1/mC_{max}^{bs}$, where C_{max}^{bs} is the length of the longest route in the best-so-far solution.

3 Computational Experiments

3.1 TSPLIB Instances

The performance of the algorithm was tested on standard TSPLIB instances, ranging from 51 to 417 cities and using integer distances. To the author's knowledge there are not many computational results available for the MTSP with MinMax objective. The found ones were both using neural network approaches to tackle this difficult problem.

The results were compared to results reported in [7] and [17]. The first one uses competition-based neural network (cNN) approach to solve exactly the same problem as presented here, the MTSP with MinMax objective. The second one is an improved version of self-organizing map (iSOM) approach based on [18] and that is modified to solve also the MTSP.

Table 1. Computational results for TSPLIB instances using integer distances.

			TACO		cNN [7]		
Problem	n	m	Best	Avg.	Best	Avg.	I%
Eil51	51	2	224	224.70	247	248.67	10
		3	159	163.00	170	172.00	5
		4	130	131.60	136	137.33	4
Eil76	76	2	278	281.00	289	292.00	4
		3	194	199.10	205	210.50	5
		4	161	163.60	159	162.75	-1
Eil101	101	2	327	330.30	340	344.67	4
		3	226	227.80	232	236.00	3
		4	178	181.00	187	189.67	5
KroA200	200	2	15376	15499.30	17353	17547.50	12
		3	10997	11186.50	11502	11722.00	5
		4	8917	9134.40	10433	10776.33	15
Fl417	417	2	6804	6962.80	7207	7266.75	4
		3	5296	5470.00	5618	5902.50	7
		4	4844	5073.80	5032	5109.50	1

			TACO		iSOM [17]	
Problem	n	m	Best sum	Avg. sum	Avg. sum	I%
Eil51	51	2	440	443.80	467	5
		3	463	469.20	494	5
		4	502	508.40	548	7
Eil76	76	2	551	558.30	584	4
		3	571	583.10	598	2
		4	618	625.80	632	1
Eil101	101	2	650	655.10	697	6
		3	670	675.30	708	5
		4	696	708.30	730	3

Solutions obtained by cNN and TACO are easily comparable, for both algorithms have the same objective. The solutions found by TACO are averagely about 6% better than of cNN. In the iSOM approach the results are given in route sums and it is not mentioned what criterion is used in evaluation. In every case, the solutions found by TACO are averagely about 4% better than the solutions of iSOM. The results of the computational experiments, including improvement percentage, are presented in Table 1.

The parameter settings were kept as standard as possible, using the values suggested in ACS, $\alpha = 1$, $\beta = 2$, $\rho = \xi = 0.1$, $N = 10$, $q_0 = 0.9$. In all runs the algorithm was allowed to run for 150 iterations, using 2-opt and 3-opt of depth 30. Candidate list of length 20 was used in all instances except Fl417, where the length was set to 100. Averages are over 10 runs.

3.2 Non-Euclidean Instances

In most real life cases the distance graphs are not Euclidean. For example, in path planning, there may be obstacles blocking the way or different kinds of terrains making the planning more complicated. Luckily most of these features can be presented by adding appropriate weigths to the distance graph. For example, when having a map of indoor environment in which some of the nodes are separated by walls, one can represent this by adding a big weight

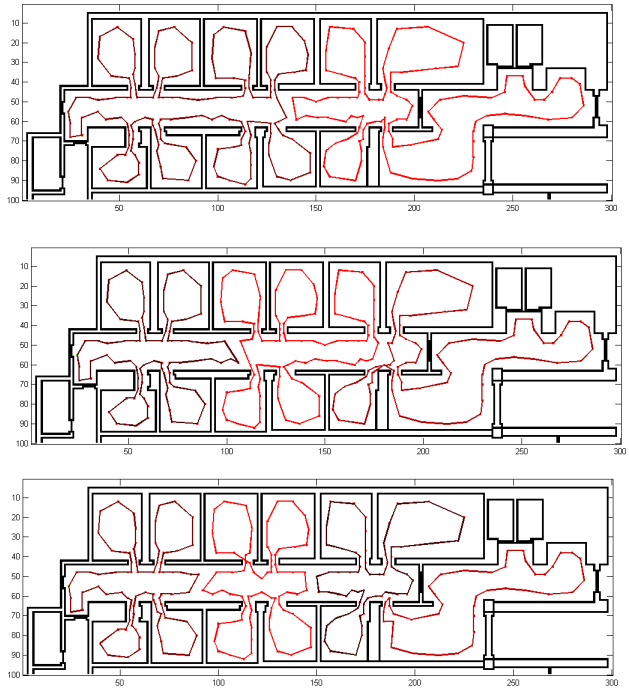

Figure 4. Sample routes for 2, 3, and 4 robots in a hospital environment

(penalty) to edges crossing a wall, so that those edges will never appear in the final solution. Similar approach is used in [5].

If there is a certain time-consuming task, such as cleaning or a measurement task, to be performed at some location, the time required to perform the given task can be taken into account by adding half of the corresponding task dependent weight to each edge connected to the node, *i.e.*, the total task weight is distributed to all possible pairs of input and output edges.

In the experiment TACO was tested on a real life case consisting of a floorplan of a hospital. The task was to plan routes for multiple robots through all checkpoints ($n = 256$). A complete graph was generated based on the floorplan, distances being otherwise Euclidean, but having a penalty weight of 1000 added to edges crossing a wall (width of the floorplan is 300). An illustration of the results generated by runs of 300 iterations can be seen in Fig. 4.

4 Conclusion

In this paper a novel team-based ant algorithm modification is proposed for the MTSP with MinMax objective. Computational results show that it can achieve good results in the TSPLIB instances using rather low number of iterations. Its general nature makes it applicable also for non-Euclidean instances, *i.e.* most real life cases. Combined with relatively easy implementation, it is a promising candidate for

robot path planning, for example.

At the time, TACO is implemented only on MATLAB causing it to be quite slow in solving bigger instances. With the parameter settings used in the experiments it takes about 60s, on Intel Core 2 CPU, 6600@2.40GHz, 2GB RAM, to run 150 iterations on Eil101. An implementation made in C or C++ should make the algorithm considerably faster and allow its use in many real life applications.

In future using other ACO algorithms, mainly MMAS, as the base for TACO should be considered. Rank-based modifications could offer more diversed search and learning than the aggressive ACS used in this paper.

References

[1] Marco Dorigo and Luca Maria Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, April 1997.

[2] E. Lawler and A. Rinnooy Kan. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. J. Wiley, New York, 1985.

[3] Gilbert Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247, June 1992.

[4] J. F. Cordeau, M. Gendreau, G. Laporte, J. Y. Potvin, and F. Semet. A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, 53:512–522, 2002.

[5] M. Kulich, J. Kubalik, J. Klema, and J. Faigl. Rescue operation planning by soft computing techniques. In *Intelligent Systems Design and Applications (ISDA 2004), August Budapest, Hungary*, 2004.

[6] P. M. Franca, M. Gendreau, G. Laporte, and F. M. Müller. The m-traveling salesman problem with minmax objective. *Transportation science*, 29(3):267–75, 1995.

[7] Samerkae Somhom, Abdolhamid Modares, and Takao Enkawa. Competition-based neural network for the multiple travelling salesmen problem with minmax objective. *Computers & OR*, 26(4):395–407, 1999.

[8] Donald Sofge, Alan Schultz, and Kenneth DeJong. *Evolutionary Computational Approaches to Solving the Multiple Traveling Salesman Problem Using a Neighborhood Attractor Schema*, pages 153–162. Springer-Verlag, London, UK, 2002.

[9] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26(1):29–41, 1996.

[10] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization (Bradford Books)*. The MIT Press, July 2004.

[11] Bernd Bullnheimer, Richard Hartl, and Christine Strauss. An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, 89:319–328, 1999.

[12] Vittorio Maniezzo and Alberto Colorni. The ant system applied to the quadratic assignment problem. *Knowledge and Data Engineering*, 11(5):769–778, 1999.

[13] T. Stützle and H. Hoos. The max-min ant system and local search for the traveling salesman problem. In *Proceedings of the Fourth International Conference on Evolutionary Computation (ICEC'97)*, pages 308–313. IEEE Press, 1997.

[14] Pan Junjie and Wang Dingwei. An ant colony optimization algorithm for multiple travelling salesman problem. In *ICICIC '06: Proceedings of the First International Conference on Innovative Computing, Information and Control*, pages 210–213, Washington, DC, USA, 2006. IEEE Computer Society.

[15] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

[16] David S. Johnson and Lyle A. McGeoch. *Local Search in Combinatorial Optimization*, chapter The Traveling Salesman Problem: A Case Study in Local Optimization, pages 215–310. John Wiley & Sons, 1997.

[17] Anmin Zhu and Simon X. Yang. An improved self-organizing map approach to traveling salesman problem. In *Proceedings of the 2003 IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*, pages 674–679, Changsha, China, October 2003.

[18] Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1995. (Second Extended Edition 1997).