Self-Organizing Map for the Prize-Collecting Traveling Salesman Problem

Jan Faigl[1] and Geoffrey A. Hollinger[2]

[1]Czech Technical University in Prague,
Department of Computer Science and Engineering,
Technická 2, 166 27 Prague, Czech Republic

[2]Oregon State University,
School of Mechanical, Industrial, and Manufacturing Engineering,
204 Rogers Hall, Corvallis, OR 97331

Abstract. In this paper, we propose novel adaptation rules for the self-organizing map to solve the prize-collecting traveling salesman problem (PC-TSP). The goal of the PC-TSP is to find a cost-efficient tour to collect prizes by visiting a subset of a given set of locations. In contrast with the classical traveling salesman problem, where all given locations must be visited, locations in the PC-TSP may be skipped at the cost of some additional penalty. Using the self-organizing map, locations for the final solution may be selected during network adaptation, and locations where visitation would be more expensive than their penalty can be avoided. We have applied the proposed self-organizing map learning procedure to autonomous data collection problems, where the proposed approach provides results competitive with an existing combinatorial solver.

1 Introduction

The self-organizing map (SOM) is a two-layered artificial neural network that can be considered as a non-linear transformation (map) of a high-dimensional input space into a lower dimensional discrete output space. Its main feature is that it preserves topological properties of the input space in the output space. Although SOM was originally proposed as a data visualization technique, it has also been applied to solve NP-hard routing problems. The first such attempt was SOM for the Traveling Salesman Problem (TSP), which was proposed in 1988 by Angéniol and Fort.

The traveling salesman problem can be formulated as follows. Having a set of locations (cities) in a plane and a distance function between them, the TSP stands to find a shortest tour connecting all the given cities, such that each city is visited exactly once and the tour returns to the origin city. This problem arises from many practical applications [2], and the TSP is a well-studied problem in the operational research domain where efficient heuristics have been proposed [12].

SOM for the TSP has the output layer organized in a one-dimensional array of units representing a Peano curve that fills the input space. During unsupervised learning, the cities are presented to the input layer, and the network is adapted by the means of moving neuron weights towards each presented city. Then, a solution is found as a sequence of cities retrieved by traversing the output layer where each winning neuron has an associated city in the input space.

Even though SOM has improved its performance in the TSP over the last decades [14, 7, 8], combinatorial heuristics still provide better results in classical graph-based instances of the TSP. On the other hand, SOM exhibits interesting results in planning problems where the locations to be visited are not explicitly prescribed [9] or where each location is represented by a set of possible points to be visited (i.e., the traveling salesman problem with neighborhoods (TSPN) [10]).

In this paper, we follow the recent advancements of SOM and propose a new adaptation rule to solve a variant of the TSP called the Prize Collecting Traveling Salesman Problem (PC-TSP) [6]. This problem is an extension of the standard TSP, where each city represents a prize that might be collected and where each prize also has an associated penalty cost if it is not collected. Thus, in the case where the penalty is significantly lower than the travel cost to the city, it is more suitable to avoid visitation of the city by the tour. The problem is to find a cost-efficient tour collecting the most important prizes (high penalty cities), i.e., to find a tour with the minimal total cost that is computed as the sum of the tour length (cost) in addition to the sum of penalties of all cities that are not visited by the tour.

The herein proposed SOM-based approach for the PC-TSP is based on the self-adjusting structure of the neural network proposed in [10] for solving the TSPN that has been extended to select and adapt the network to the most promising cities. To the best of our knowledge, the proposed method is the first application of SOM to the PC-TSP. The approach extends the application domain of SOM to additional optimization problems in which SOM can provide new ways of solving routing problems. The PC-TSP represents a class of problems where it is not sufficient to just alternate cities, but where it is also desirable to learn the underlying problem domain to select the most important cities to visit. Finding a solution to the PC-TSP consists of 1) a selection of the most promising cities and those that should be avoided and 2) finding a tour visiting the selected cities. SOM can be used to simultaneously address both of these problems together.

The paper is organized as follow. The problem motivation and the problem definition are presented in the next section. The proposed method is introduced in Section 3. Results of the proposed approach evaluation and a comparison with other methods are presented in Section 4 together with a discussion of found insights. Section 5 is dedicated to concluding remarks.

2 Problem Statement

The addressed problem is motivated by autonomous data collection, where it is requested to collect data from a number of sampling stations to create a model of a spatial phenomena as quickly as possible. Due to the spatial distribution of the sampling stations, the information retrieved from one station can also be included in measurements provided by other stations; thus, it is not necessary to retrieve data from all stations to acquire the desired model of the phenomena. The problem is formulated as a simplified variant of the PC-TSP [5], where each city has an associated penalty representing an importance of the measurement provided by the station [13].

In robotics, the TSP-based routing problems are alternatively called multi-goal path planning problems because the cities in the TSP represent goals towards which the robot navigates [13]. Therefore, to emphasize the robotic motivation of the studied problem, the term goal (or goal location) is used in the rest of this paper to denote the equivalent of cities in the TSP.

2.1 Problem Definition

Having n possible goal locations $\mathbf{G} = \{g_1, \dots, g_n\}$, $g_i \in \mathbb{R}^2$, where each goal has associated penalty $\zeta(g_i) \geq 0$, and distance between two goals g_i and g_j is $c(g_i, g_j) \geq 0$, the problem is to find a tour T visiting a subset of the goals $G_T \subseteq \mathbf{G}$ such that the total cost of the tour $\mathcal{C}(T)$ is minimal

$$\mathcal{C}(T) = \sum_{(g_{s_i}, g_{s_{i+1}}) \in T} c(g_{s_i}, g_{s_{i+1}}) + \sum_{g \in \mathbf{G} \setminus G_T} \zeta(g). \tag{1}$$

The tour T is a sequence of the selected goals $T = (g_{s_1}, \dots, g_{s_{k-1}}, g_{s_k})$, where $g_{s_j} \in G_T$, $s_j \geq 1$, and $s_j \leq n$. The sequence represents a closed tour over the selected goals $g_{s_1} = g_{s_k}$, and all the selected goals, except the first (and last) visited goal, are included in the tour at most once.

For simplicity, which is also in line with the considered motivational application of autonomous data collection [13], the travel cost between two goals is computed as the Euclidean distance $c(g_i, g_j) = |(g_i, g_j)|$; i.e., the problem is considered in a planar environment without obstacles.

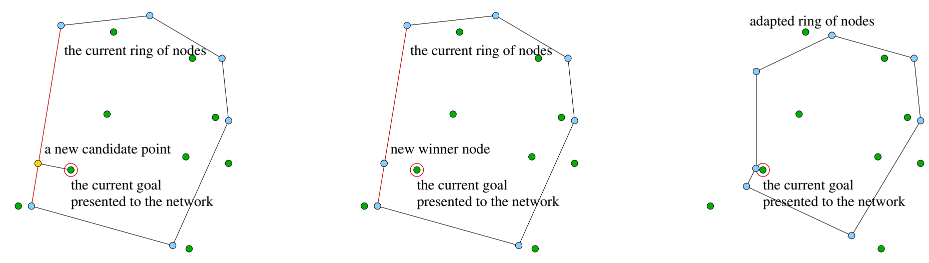
2.2 Performance Metric

The PC-TSP is NP-complete because it includes the TSP for very high penalties. Several approximation algorithms have been proposed for variants of the problem [4]. Although these algorithms provide guaranteed approximation factors relative to optimal, the approximation factors are relatively high (e.g., factor of 2.5 for the approach [6] based on the Christofides' algorithm) or somewhat better for a more complex algorithm [3].

In this paper, we consider a solution quality metric using the ratio of the PC-TSP solution to the related TSP (i.e., the shortest tour visiting all the goals and thus with zero penalty term in (1)). A solution to the TSP is available out-of-the-box using the Concorde solver [1], which provides an optimal solution for TSP problems (up to several hundred goals) within a reasonable time. Moreover, this ratio allows us to aggregate results for different problem instances and consider statistical evaluation of the algorithmic performance using several hundreds of trials. The ratio is computed as

$$R = \frac{\mathcal{C}(T)}{\mathcal{C}(T_{TSP})}, \tag{2}$$

where T_{TSP} is the optimal solution of the underlying TSP.



(a) (b) (c)

Fig. 1. Visualization of winner selection: (**a**) the closest point of the ring's segment to the presented goal; (**b**) new neuron added to the ring at the position of the closest point; (**c**) the current ring after adaptation of the winner and its neighbouring nodes. The goals are represented by green discs, while the blue discs denote the neuron weights.

3 Proposed SOM for the PC-TSP

The proposed adaptation schema for the PC-TSP is based on two relatively straightforward properties that are not included in standard SOM-based approaches for the TSP. The first is a mechanism to adjust the number of neurons according to the currently selected number of goals to be visited. Otherwise, a poor solution will result because a low number of neurons will not provide convergence guarantees, and too many neurons will result in an inefficient adaptation of the winner neighborhood. The second property is the selection of the goals itself.

The first property is addressed by considering the self-adjusting adaptation rules proposed in [10]. The selection of goals is addressed in winner selection, where the neuron is considered to be a winner candidate only if its distance to the goal currently presented to the network is smaller than the goal's penalty. These two ideas are the foundation of the proposed solution.

The learning procedure consists of a two-layered competitive neural network similar to the one used for the TSP. The input layer represents a two-dimensional input vector, and the second layer consists of the output units organized into a uni-dimensional structure, where the neuron weights represent coordinates in a plane. The output units are defined by a sequence of straight line segments (called the ring) in $\mathbb{R}^2$ that represents the tour connecting the selected goals.

The main idea of the self-adjusting neural network is based on winner selection using the shortest distance of the presented goal to the network (i.e., the shortest distance of the goal to the ring). Having the current ring of neuron weights as a sequence of the straight line segments, the winner candidate is found as the closest point of the ring to the presented goal. If such a point corresponds to a neuron already presented in the ring, and the neuron was not a winner in the current learning epoch, the neuron is considered the winner candidate. Otherwise, a new neuron is created, and its weights are set to the position of the point. If such a winner candidate becomes the winner, the winner neuron is added to the ring; otherwise it is deleted whenever a new winner candidate is determined. The winner selection is schematically visualized in Fig. 1.

The winner selection provides a mechanism to create new neurons. The reduction of neurons is performed by the ring regeneration at the end of each learning epoch. At this time, the current winner neurons are preserved, and all other neurons are removed from the network. An additional neuron is inserted between each preserved winner to support spreading neurons during the adaptation. The weights of the neurons are set to the center of the segment connecting the two winners. The learning procedure is summarized in the following eight steps:

1. *Initialization:* Create $2n$ neurons around the first goal for the input set of n goals $\mathbf{G}$. Set the neighbouring function variance $\sigma \leftarrow 10$ and the learning rate $\mu \leftarrow 0.99$. Set the current number of learning epoch $i \leftarrow 1$.
2. *Randomizing:* Create a random permutation of goals $\Pi(\mathbf{G}) \leftarrow \text{permute}(\mathbf{G})$.
3. *Winner Selection:* `If` $i == 1$
   - `Then`: Select the closest point p of the current ring to the presented goal $g \in \Pi(\mathbf{G})$;
   - `Otherwise`: Select the closest point p of the current ring to $g \in \Pi(\mathbf{G})$ such that $|(p, g)| < \zeta(g)$.
4. *Adapt:* `If` p is selected
   - Determine the appropriate winner neuron $\nu^\star$ (select or create new one).
   - Adapt the winner $\nu^\star$ and its neighbouring nodes ν_j within the distance d (in the number of nodes) using the neighbouring function $f(\sigma, d) = \mu e^{(-d^2/\sigma^2)}$ for $d < 0.2m$ and $f(\sigma, 0) = 0$ otherwise, where m is the current number of neurons, i.e., move ν closer towards g about $|(\nu, g)|f(\sigma, d)$.

   Remove g from the permutation, $\Pi(\mathbf{G}) \leftarrow \Pi(\mathbf{G}) \setminus \{g\}$, and `If` $|\Pi(\mathbf{G})| > 0$ go to Step 3.
5. *Ring regeneration:* Create a new ring using only the winner nodes of the current learning epoch, and add a new neuron between each two consecutive winner nodes ν_i and ν_j with the weights set to $\nu_i + (\nu_i - \nu_j)/2$.
6. *Update the number of the learning epoch and neighbouring function variance:* $i \leftarrow i + 1$; $\sigma \leftarrow (1 - 0.0005i)\sigma$.
7. *Termination condition:* `If` the maximal distance of the winner to its goal is less than 10^{-3}, stop the adaptation. `Otherwise` go to Step 2.
8. *Final tour construction:* Traverse the ring and use the associated goals to the last winners to construct the final goal tour.

Computational complexity of the learning procedure can be derived from the number of comparisons needed to find the best matching neuron for each presented goal to the network, which is the most time-consuming operation. The number of goals is n, and the number of neurons can be bounded by $3n$ at every moment of the adaptation, which gives $3n^2$ operations. The values of μ and $f(\sigma, d)$ are always less than 1, and thus the adaptation rule is stable [15]. Besides, the neighbors of the winner are effectively moved only for a sufficiently high value of $f(\sigma, d)$, and since σ does not depend on n, the network is stabilized in a constant number of learning epochs. Thus, the overall computational complexity can be bounded by $O(n^2)$.

The required memory only depends on the representation of the goals and neurons, which are basically coordinates in the plane accompanied by the penalty and associated goal for the winners. Hence, the space can be bounded by $O(n)$.

4 Results

The proposed SOM for the PC-TSP was first validated in simple use cases to verify feasibility of the proposed principle of goal selection. Then, its performance was evaluated in problem instances proposed in [13]. Results from these two evaluation scenarios are presented in the following sections.

4.1 Simple Use Cases

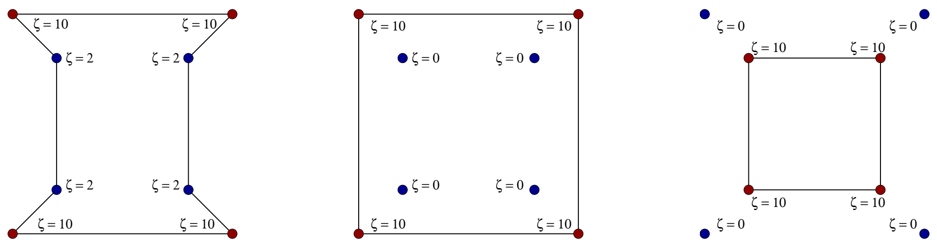


Fig. 2. Example of found solutions for concentric squared goals problems.

The initial feasibility test was performed using 8 goals forming two concentric squares with a side length 10 and 8. According to the value of the penalties, the solution of the PC-TSP is a tour connecting all the goals (for penalties 10 and 2) and a single connected square of the outer (penalties 10 and 0) or inner (penalties 0 and 10) goals. The found solutions are visualized in Fig. 2, where the red discs denote goals with higher penalties (more important goals), and goals with zero or small penalties are shown in blue. The solutions of these simple problems correspond to optimal solutions and thus provide validation of the proposed adaptation rules.

4.2 Performance Evaluation

The performance evaluation is based on the solution to data collection problems consisting of 100 randomly placed sensors within a 100 km × 100 km large area, where each goal penalty is randomly drawn from the range 0 to 25. Similarly to [13], the vehicle speed is assumed to be 5 km per hour. Thus, the goals are effectively placed in a 20 × 20 large square, and the cost between goals is directly computed as their Euclidean distance. In addition, more problem instances are created from this setup by dividing the penalty by the value f, which allows

Table 1. Average ratios R of the solution cost to the optimal solution of the related TSP for 50 problems in the 100 km × 100 km area and 2500 SOM trials

Problem	f	**SOM TSP** [10]	**PC-TSP** [11],[1]	**SOM PC-TSP** Proposed	**SOM PC-TSP+TSP** Proposed + [1]
area 20×20	0.001	1.03/0.02	1.00/<0.01	1.04/0.02/(−)	**1.00/<0.01/(+)**
area 20×20	0.100	1.03/0.02	1.00/0.01	1.03/0.02/(−)	**1.00/<0.01/(+)**
area 20×20	1.000	1.03/0.02	1.02/0.02	1.02/0.02/(=)	**0.99/0.01/(+)**
area 20×20	2.000	1.03/0.02	1.02/0.03	**1.01/0.02/(+)**	**0.98/0.02/(+)**
area 20×20	5.000	1.03/0.02	1.03/0.05	**0.97/0.05/(+)**	**0.95/0.05/(+)**
area 20×20	7.000	1.03/0.02	1.00/0.04	1.00/0.07/(=)	0.99/0.07/(=)
area 20×20	10.000	1.03/0.02	0.84/0.08	**0.79/0.06/(+)**	**0.79/0.06/(+)**

Values in columns are: average / standard deviation / (statistical comparison).
(+) - the algorithm provides statistically better solutions than the PC-TSP [11, 1].

us to study the algorithm's performance for different penalties. Notice, that for very high penalties (e.g., $f \leq 0.1$), the problems become close to the TSP.

The proposed SOM approach is compared with the combinatorial deterministic approach considered in [13]. The prior combinatorial approach was based on the heuristic determination of the goals to ignore [11] and a consecutive optimal solution of the TSP for the remaining goals using the Concorde solver [1]. The SOM approach is considered in two variants. First, it is used for simultaneous selection of the goals together with the tour connecting them. The second variant uses the goals determined by the first variant that are connected by the optimal tour found by [1].

For each problem scenario, 50 random instances are created, and the algorithm's performance is measured as the average value of the ratio R of the found solution of the PC-TSP to the optimal solution of the related TSP introduced in (2). Due to stochastic nature of SOM, 50 trials are solved for each problem instance, which gives 2500 solutions to compute the average ratio for a particular problem scenario.

Based on the statistical data, a comparison of the SOM-based algorithms with the reference algorithm was performed. This tests the null hypothesis that the random variables R describing the quality of the provided solutions are from the some distribution. For each scenario and particular factor, the ratio R is considered to be a random variable over the 50 problem instances. For SOM, the average C from all 50 trials is considered to be the solution quality of the scenario. These random variables are considered to be drawn from normal distributions since the problems are random. The algorithms are considered to provide statistically different results if the P-value of the T-test is below 0.05. In this case, the algorithm with a lower average value of R is considered better (denoted by the character '+'). The computed averages, standard deviations, and results of statistical comparison are presented in Table 1.

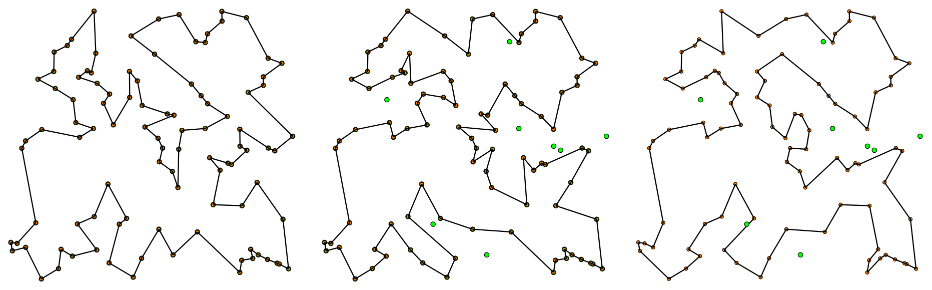
The results in Table 1 indicate that SOM solutions of the PC-TSP provide paths with almost identical costs as the pure solution of the TSP. Even though not all sensors are visited, the cost of the final path is (on average) similar to the tour visiting all the goals. To further explore the benefit of the proposed approach, an additional set of problems was created within a larger area with

Table 2. Average ratios R of the solution cost to the optimal solution of the related TSP for 50 problems in the 200 km $\times$ 200 km area and 2500 SOM trials

Problem	f	**SOM TSP** [10]	**PC-TSP** [11],[1]	**SOM PC-TSP** Proposed	**SOM PC-TSP+TSP** Proposed + [1]
area 40×40	0.001	1.03/0.02	1.00/<0.01	1.04/0.02/(−)	1.00/<0.01/(=)
area 40×40	0.100	1.03/0.02	1.00/<0.01	1.04/0.02/(−)	1.00/<0.01/(=)
area 40×40	1.000	1.03/0.02	0.98/0.02	1.00/0.03/(−)	**0.95/0.02/(+)**
area 40×40	2.000	1.03/0.02	0.98/0.02	0.97/0.03/(=)	**0.95/0.02/(+)**
area 40×40	5.000	1.03/0.02	0.91/0.06	**0.78/0.05/(+)**	**0.78/0.05/(+)**
area 40×40	7.000	1.03/0.02	0.68/0.12	**0.58/0.03/(+)**	**0.58/0.03/(+)**
area 40×40	10.000	1.03/0.02	0.42/0.06	0.43/0.03/(=)	0.43/0.03/(=)

average value / standard deviation / (statistical comparison)
(+) - the algorithm provides statistically better solutions than the PC-TSP [11, 1].

dimensions 200 km $\times$ 200 km, which provides a better opportunity to avoid distant goals with small penalties. Statistical indicators for these problems are presented in Table 2, and an example of found solutions is depicted in Fig. 3.

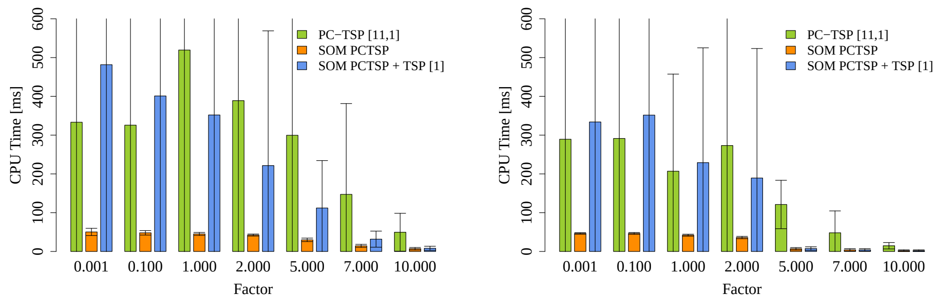


(a) SOM TSP, $\mathcal{C}$=316 (b) SOM PC-TSP, $\mathcal{C}$=309 (c) SOM PC-TSP+TSP, $\mathcal{C}$=297

Fig. 3. Found solutions by the SOM for the TSP and PC-TSP.

Regarding the required computational time, the proposed SOM algorithm provides solutions of the PC-TSP (with 100 goals) in less than ten milliseconds using a single core of the iCore7 processor running at 3.4 GHz. The real computational requirements are depicted in Fig. 4. In some cases, finding the optimal solution of the TSP by [1] is computationally demanding, which makes standard deviations very high. On the other hand, the proposed SOM based PC-TSP algorithm requires almost identical computational time per particular problem scenario. The computational burden decreases for lower penalties, which is caused by selection of only few goals to be visited.

Discussion – The presented results indicate the proposed SOM adaptation rules provide a feasible solution to the PC-TSP that achieves performance competitive with other approaches. Moreover, the results also indicate that for problems where the selection of the goals is more important (i.e., problems with a higher penalty factor f), the SOM approach performs better.



(a) area 20 × 20 (b) area 40 × 40

Fig. 4. Real computational time (in milliseconds) to solve the PC-TSP.

On the other hand, the benefit of solving the considered instances using the PC-TSP is not always evident because the solution quality is sometimes similar to the solution of the pure TSP. This is mainly caused by instances of the problem with dense goals, as is indicated by the results for goals placed within a larger area. Similar findings have been reported in [13]. All together, these results indicate that solving the motivational data collection problem as an instance of the PC-TSP provides substantial benefit when the penalties are small relative to the size of the environment.

The addressed problem provides the groundwork for studying the principles of SOM adaptation employed in routing problems. Here it is worth mentioning that an explicit consideration of the final cost of the tour represented by the ring in the selection of winners does not provide good results, and the convergence of the network becomes much slower than for the proposed approach. This also holds for different adaptation schemata (e.g., a fixed number of neurons), which do not provide stable adaptation with competitive results.

5 Conclusion

Novel adaptation rules to address the prize-collecting traveling salesman problem using a self-organizing map have been introduced in this paper. The main feature of the proposed SOM for the PC-TSP is that it provides a simultaneous selection of the goals together with a sequence of their visit. The proposed method uses SOM-based solution to the PC-TSP that provides competitive results to other methods. As such, it extends the portfolio of routing problems where SOM can be applied.

In the motivational autonomous data collection task, the PC-TSP provides substantial reduction in solution cost when the penalties are small relative to the size of the environment. Previous research indicates that considering a non-zero communication radius to read the data from the sensor station from a longer distance can improve the solution more significantly. This extension would lead to a combination of the PC-TSP with the traveling salesman problem with neighborhoods (TSPN), where SOM has also been successfully applied. Therefore, we plan to extend the proposed approach to address such combined problems. In addition, our future work is also directed towards considering dynamic evaluation of the penalty when it depends on both the current tour and the expected goals to be visited.

Acknowledgments

The presented work is supported by the Czech Science Foundation (GAČR) under research project No. 13-18316P. Support under United States National Science Foundation Grant IIS-1317815 to Geoffrey Hollinger is also gratefully acknowledged.

References

1. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: CONCORDE TSP Solver (2003), `http://www.tsp.gatech.edu/concorde.html`, [cited 20 Oct 2013]
2. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: The Traveling Salesman Problem: A Computational Study. Princeton University Press, Princeton, NJ, USA (2007)
3. Archer, A., Bateni, M.H., T., H.M., Karloff, H.: Improved approximation algorithms for prize-collecting steiner tree and tsp. In: IEEE Symposium on Foundations of Computer Science (2009)
4. Ausiello, G., Bonifaci, V., Leonardi, S., Marchetti-Spaccamala, A.: Prize-collecting traveling salesman and related problems. In: Gonzalez, T.F. (ed.) Handbook of Approximation Algorithms and Metaheuristics. CRC Press (2007)
5. Balas, E.: The prize collecting traveling salesman problems. Networks 19, 621–636 (1989)
6. Bienstock, D., Goemans, M., Simchi-Levi, D., Williamson, D.: A note on the prize collecting traveling salesman problem. Mathematical Programming 59, 413–420 (1993)
7. Cochrane, E.M., Beasley, J.E.: The co-adaptive neural network approach to the Euclidean travelling salesman problem. Neural Networks 16(10), 1499–1525 (2003)
8. Créput, J.C., Koukam, A.: A memetic neural network for the Euclidean traveling salesman problem. Neurocomputing 72(4-6), 1250–1264 (2009)
9. Faigl, J.: Approximate Solution of the Multiple Watchman Routes Problem with Restricted Visibility Range. IEEE Transactions on Neural Networks 21(10), 1668–1679 (2010)
10. Faigl, J., Přeučil, L.: Self-Organizing Map for the Multi-Goal Path Planning with Polygonal Goals. In: ICANN. pp. 85–92 (2011)
11. Goemans, M., Williamson, D.P.: A general approximation technique for constrained forest problems. SIAM J. Computing 24(2), 296–317 (1995)
12. Helsgaun, K.: An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. European Journal of Operational Research 126(1) (2000)
13. Hollinger, G., Mitra, U., Sukhatme, G.: Autonomous data collection from underwater sensor networks using acoustic communication. In: IROS. pp. 3564–3570. IEEE (2011)
14. Somhom, S., Modares, A., Enkawa, T.: A self-organising model for the travelling salesman problem. Journal of the Operational Research Society pp. 919–928 (1997)
15. Tucci, M., Raugi, M.: Stability analysis of self-organizing maps and vector quantization algorithms. In: IJCNN. pp. 1–5 (2010)