On the Performance of Self-Organizing Maps for the non-Euclidean Traveling Salesman Problem in the Polygonal Domain

Jan Faigl

Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, 166 27, Prague 6, Czech Republic

Abstract

In this paper, two state-of-the-art algorithms for the Traveling Salesman Problem (TSP) are examined in the multi-goal path planning problem motivated by inspection planning in the polygonal domain $\mathcal{W}$. Both algorithms are based on the self-organizing map (SOM) for which an application in $\mathcal{W}$ is not typical. The first is Somhom's algorithm, and the second is the Co-adaptive net. These algorithms are augmented by a simple approximation of the shortest path among obstacles in $\mathcal{W}$. Moreover, the competitive and cooperative rules are modified by recent adaptation rules for the Euclidean TSP, and by proposed enhancements to improve the algorithms' performance in the non-Euclidean TSP. Based on the modifications, two new variants of the algorithms are proposed that reduce the required computational time of their predecessors by an order of magnitude, therefore making SOM more competitive with combinatorial heuristics. The results show how SOM approaches can be used in the polygonal domain so they can provide additional features over the classical combinatorial approaches based on the complete visibility graph.

Keywords: Self-Organizing Map (SOM); Traveling Salesman Problem (TSP); Multi-Goal Path Planning

1. Introduction

The self-organizing map (SOM) also known as Kohonen's unsupervised neural network, was first applied to the Traveling Salesman Problem (TSP) by Angéniol [1] and Fort [19] in 1988. The TSP is probably the most famous combinatorial problem studied by the operational research community for more than five decades [10]. The problem is to find a route for visiting a given set of n cities (goals) so that the length of the route is minimized. In SOM, the output neurons are organized into a unidimensional structure (cycle), and a solution is represented by synaptic weights that are adapted to the cities during the self-adaptation process. After the adaptation, the neurons are associated to the cities, and because of the unidimensional structure, the final city tour can be retrieved by traversing the cycle.

The SOM adaptation schema for the TSP consists of two phases. A city is presented to the network, and a winner neuron is selected in the *competitive* phase. For a planar TSP where cities represent points in $\mathcal{R}^2$, the neurons' weights can be considered as points in the plane that are called nodes in this paper. So, the winner neuron is the node with the smallest distance to the city. Then, the adaptation can be described as a movement of the winner node together with its neighboring nodes toward the city. The adaptation is called a *cooperative* phase, as neighboring nodes also move, although by a shorter distance. After the complete presentation of all cities (one adaptation step), the procedure is repeated until the termination condition is not met, e.g., when the winner nodes are sufficiently close to the cities.

Several SOM approaches have been proposed [3, 4, 6–9, 11, 34–36] in the history of the SOM application to the TSP. In these approaches, the adaptation rules have been modified [37, 39], heuristics have been considered [30], and combinations with genetic algorithms [26], memetic [12] or immune system [27] approaches have been proposed. Even though these new approaches improve the performance of SOM for the TSP, SOM is still not competitive with the combinatorial approaches to the TSP in both aspects: the solution quality and required computational time [12]. It should also be noted that all of the above mentioned SOM approaches consider only the Euclidean variant of the TSP, i.e., distances between cities are Euclidean, while combinatorial approaches generally work with graphs.

Herein, the TSP is considered in the context of inspection planning, where cities represent sensing locations in the polygonal domain $\mathcal{W}$ [17, 21]. The problem is to find a path for a mobile robot so that the robot will "see" the whole

working space. The practical motivation of the problem is a search and rescue mission in which possible victims need to be found quickly [25]. The problem can be formulated as the TSP, i.e., a problem to find a path that visits the given set of sensing locations, where measurements of the robot's vicinity are taken. The robot's working space is represented by a polygonal map that may contain obstacles, therefore collision-free paths among obstacles (geodesic paths) have to be considered [32]. It is also the case of the SOM adaptation procedure where the geodesic paths (distances) between nodes and cities have to be considered rather than Euclidean distances, otherwise a poor solution would be found. The node–city distances are used in the competitive phase, in which a winner node is selected. Shortest paths are then used in the cooperative phase where nodes are adapted toward a city along a particular path, i.e., the node is placed at a new position on the path closer to the city.

It is clear that a determination of the shortest path among obstacles is more computationally intensive than a direct usage of the Euclidean distance. From this perspective, the complexity of SOM algorithms increase in $\mathcal{W}$ because the adaptation rule has to be augmented by an algorithm to find geodesic paths. However, combinatorial approaches based on a graph problem representation can be directly used in $\mathcal{W}$ without any modifications. The costs of edges between cities are lengths of the shortest paths between cities, and the graph can be constructed from the visibility graph by Dijkstra's algorithm. Therefore, the gap between SOM and combinatorial heuristics seems to be wider for the TSP in the polygonal domain.

In [18], a simple, yet sufficient approximation of the shortest path in $\mathcal{W}$ has been used in SOM adaptation rules to decrease the computational burden. Although this approximation enables the application of SOM principles in $\mathcal{W}$, the required computational time of self-organization is still significantly higher (hundreds of times) than for the Euclidean-TSP. The main issue of the conventional SOM is the high number of node–city distance queries in the competitive phase, and also the relatively high number of node–city path queries in the cooperative phase. In this performance study, the computational requirements of these adaptation phases are examined in two different ways. First, technical aspects of the queries are considered, i.e., the required computational time is reduced by informing the competitive selection procedure and assuming practical approximations in the cooperative phase. The second way aims to decrease the number of queries considering recent adaptation rules; these reduce the required number of adaptation steps, and effectively decrease the size of the winner node neighborhood. The rules are closely related to the initialization of adaptation parameters; therefore, various initializations are considered as well.

The rest of the paper is organized as follows. Section 2 describes the notation and terminology used related to the geometrical structures supporting the shortest path queries and adaptation procedure. Related work is acknowledged in Section 3. In Section 4, a brief description of the SOM adaptation procedures used herein and an approximation of the shortest path in the polygonal domain $\mathcal{W}$ is presented. Section 5 presents proposed modifications and combinations of published competitive and cooperative rules to decrease the required computational time possibly without affecting the solution's quality. Experimental results are presented in Section 6. Finally, the concluding remarks are presented in Section 7.

2. Terms Used and Notation

The SOM adaptation is considered in the polygonal domain, i.e., a polygonal map; therefore, a few terminology notes are presented in this section to clarify the terms used and symbols for supporting geometrical structures.

A world is represented by a polygonal map $\mathcal{W}$ consisting of N_V vertices; thus, $\mathcal{W}$ is a closed, multiply connected region, whose boundary is a union of N_V line segments, forming $h+1$ closed polygonal cycles, where h is the number of holes (obstacles). A distance between two points inside $\mathcal{W}$ is a length of a path among obstacles that can be a straight line segment or consisting of vertices. Thus, a path between two points s and t consists of a finite number of line segments joining the points and vertices.

$\mathcal{W}$ can be divided into a set of non-overlapping convex polygons that are formed from vertices. Such convex polygons are called cells and represent a *convex polygon partition* of $\mathcal{W}$, i.e., each cell C forms a closed polygonal cycle of line segments joining vertices. A line segment is called *diagonal* if it connects two nonadjacent vertices and if it is contained in $\mathcal{W}$. A point inside $\mathcal{W}$ is always inside a cell, and a collision-free path between two points $s \in C_s$ and $t \in C_t$ can be constructed from the shortest path between vertices of C_s and C_t. Weights of the i*th* neuron represent a point v_i (called node) that lies in $\mathcal{W}$; therefore, v_i is always inside a cell. Such a cell of the node v is denoted as C_v. An example of a polygonal map, its convex partition, and a path from a node to a city is shown in Fig. 1.

The symbols used are as follows.

$\mathcal{W}$	the polygonal domain representing the working space to be inspected, $\mathcal{W} \subset \mathbb{R}^2$
N_V	the number of vertices of $\mathcal{W}$
n	the number of cities
m	the number of nodes (neurons)
$\|s,t\|$	the Euclidean distance between points s and t
$\|S(v_i, v_j)\|$	the length of the shortest path (among obstacles) between two vertices v_i and v_j in $\mathcal{W}$
$\boldsymbol{P}$	a set of convex polygons
v_i	a vertex of the polygonal domain $\mathcal{W}$
ν_i	a node representing the weights of the i^{th} neuron
C_ν	the cell of $\boldsymbol{P}$ in which node ν lies
G	the learning gain (also called the neighborhood function variance)
μ	the learning rate
α	the gain-decreasing rate
d	the number of neighboring nodes of the winner node

3. Related Works

The first application of SOM principles to the TSP [1] follows constructive heuristics and starts with one node. In that approach, a node is duplicated if it is the winner for two different cities, and it is deleted if it is not selected as the winner for three complete presentations of cities to the network. Growing ring structure has also been used in FLEXMAP proposed in 1991 [20]; however, the deletion mechanism was omitted. The maximal number of required nodes has been close to $2.5n$, where n is the number of cities, up to a problem with 2392 cities. In [6], Budinich used the same number of nodes as cities, and the inhibition was replaced by a real value derived from the winner node and its neighboring nodes. The tour is constructed from an ordered sequence of cities according to the value.

An inhibition of frequently selected winner nodes has been used in the Guilty net algorithm [9]. The inhibition mechanism was substituted by the vigilance parameters in the Vigilant Net presented in [7] where the initialization of weights is discussed. Superior results are reported for starting positions of nodes as the convex hull approximation of the cities. Aras used the geometrical properties of the connected nodes forming a ring and the topology of cities in his KNIES algorithm [3]. This algorithm uses a regular adaptation of the winner node, which moves toward the city. In addition, nodes that are not in the activation bubble (set of neighboring nodes), do not move closer to the city, but move in a way that allows the preservation of the global statistical properties of the data points. The proposed algorithm has been used to solve large TSP instances by the decomposition of the problem into several clusters [4].

Probably the most complex and powerful SOM algorithm for the TSP is the Co-Adaptive net introduced in [11]. This algorithm uses a higher number of nodes than the number of cities, and it utilizes an adaptive neighborhood of the winner node that is updated after each adaptation step. The learning process is divided into competition and co-operation phases. The co-operation phase is based on winner nodes or their neighbors not moving more than once. The algorithm also uses the near-tour to tour construction, which creates a complete tour if the current winner nodes are not distinct. The best tour is kept during the adaptation, and it is used if the final solution is worse. The authors presented a huge set of results and comparisons with other approaches, and reported that their approach outperforms other variants and together with [36] provides better results than Aras's KNIES [3], which uses the statistical properties of the data points.

Even though the statistical approach (KNIES) has been outperformed, the convex hull property has been studied in the expanding SOM variant called ESOM [26]. To follow the convex hull property and to design an appropriate learning rule that considers the global parameters of the problem, Intergrated SOM (ISOM) uses an evolutionary principle and combines SOM with a genetic algorithm [22]. The convex hull property is also studied in [38], where the authors considered a more conservative learning rule than ESOM: the movement of the nodes, which follows the expansion to preserve the convex hull property is restricted. This algorithm provides almost identical results as those of ESOM, but the learning rule is much simpler.

Another research direction studied is the initialization of neuron weights and the setting of adaptation parameters: the learning rate μ, the learning gain G, and the gain-decreasing rate α. An initialization of weights was studied

in [5], where authors examined four initialization methods: random, small circle around centroid of the cities, a tour found by the nearest neighborhood algorithm, and a random initialization of nodes on a rhombic frame located to the right of the cities' centroids. The fourth initialization method is reported as the most suitable technique. Kohonen's exponential evolution of the adaptation rules is studied in [37]. To reduce the number of parameters, the authors proposed simplified adaptation rules based only on the number of performed adaptation steps k. The learning rate is defined as $\mu = 1/\sqrt[3]{k}$ and the learning gain as $G = G(1 - 0.01k)$ with initial value $G_0 = m/32$, where m is the number of neurons. They used initial weights representing nodes on a rectangular frame around the cities, and the authors reported superior results in the selected TSPLIB [31] instances in comparison with the SOM approaches [1, 19] and [9]. These simplified rules have also been applied in [39], where the authors proposed to use $\mu = 1/\sqrt[4]{k}$ and the initial value of the gain $G_0 = 10$. For small values of G, the value of the neighborhood function is very small; thus, the neighboring nodes are negligibly moved. Considering this fact, the authors recommended to gradually decrease the neighborhood of the winner node after each adaptation step. It decreases the computation burden while not affecting the quality of solution. The recommended initial size of the neighborhood is $d = 0.4m$, which is decreased by $d = 0.98d$ at the end of each adaptation step.

In [28], Murakoshi and Sato applied multiple scale neighborhood functions to decrease topological defects that may occur during the self-adaptation. The functions have a form

$$f(G, l) = \beta_j \mu e^{-\frac{l^2}{(\gamma_j G)^2}}, \tag{1}$$

where β_j and γ_j are the gain and width factors of the jth neighborhood function. The authors used six functions $\beta_j = 2^{-|3-j|}$ and $\gamma_j = 2^{-(3-j)}$ for $j \in \{1, 2, \ldots, 6\}$. These functions have been incorporated into the SOM adaptation procedure [1], where a function has been randomly chosen during the adaptation. The authors reported up to 42.65 % less kinks than in the original version of the procedure [1].

4. Self-organizing maps for the Traveling Salesman Problem

Two state-of-the-art SOM adaptation procedures are considered in this performance study as the primal algorithms being modified. The SOM algorithms have been selected regarding the results presented in [11], where these approaches provide the best performance. Although more recent approaches increase the quality of solution, the improvement is not significant, and such approaches are also more computationally demanding. That is why Somhom's algorithm introduced in [36] and the Co-adaptive net algorithm [11] have been selected for evaluation of their performance for problems in the polygonal domain $\mathcal{W}$. The performance of these algorithms is improved by the proposed modifications and by the combination of selected approaches (briefly described in the previous section), and therefore, the original algorithms are described in more detail in the next subsections. Moreover, an overview of the used shortest path approximation in $\mathcal{W}$ is presented in Section 4.3.

4.1. Somhom's Algorithm

The algorithm presented in [36] by Somhom et al. uses an inhibition mechanism, i.e., a neuron can be a winner only for one city during a single adaptation step. In the rest of this paper, Somhom's algorithm is denoted as SME. The basic schema of the algorithm is similar for both SOM algorithms considered. The schema is shown in Algorithm 1.

The algorithm works as follows. The ring of nodes is initialized as a small ring around one of the cities. The adaptation procedure consists of a sequence of adaptation steps in which all cities are randomly presented to the neural network. For each presented city, the winner node is selected according to $v^\star = \text{argmin}_v |c, v|$, where $|.,.|$ denotes the Euclidean distance between the city c and the node v for the Euclidean TSP. The adaptation procedure (`adapt`) moves the winner node and its neighboring nodes toward the presenting city c according to the rule $v'_j = v_j + \mu f(G, l)(c - v_j)$, where μ is the learning rate. The neighboring function is $f(G, l) = exp(-l^2/G^2)$ for $l < d$ and $f(G, l) = 0$ otherwise, where G is the gain parameter, l is the distance in the number of nodes measured along the ring, and d is the size of the winner node neighborhood that is set to $d = 0.2m$, where m is the number of nodes. The initial value of G is set proportionally to the problem size $G_0 = 0.06 + 12.41n$. The values of learning and decreasing rates are $\mu = 0.6$ and $\alpha = 0.1$, respectively.

Algorithm 1: Self-organizing map for the TSP

Input: $\boldsymbol{C} = \{c_1, \ldots, c_n\}$ - a set of cities
Input: (d, G, μ, α) - the parameters of SOM
Input: δ – the maximal allowable error
Input: i_{max} – the maximum number of adaptation steps
Output: $(\nu_1, \ldots, \nu_M)$ - a sequence of node weights representing city tour

```
init(ν_1, ..., ν_M)                                   // an initial set of neurons weights
i ← 0                                                 // the adaptation step counter
repeat
    error ← 0
    I ← ∅                                             // the set of inhibited nodes
    Π(C) ← a random permutation of cities
    foreach c ∈ Π(C) do
        ν* ← select winner(node to c), ν* ∉ I)        // call the select winner procedure
        error ← max{error, |ν*, c|}
        adapt(ν*, c)                                  // call the adapt procedure
        I ← I ∪ {ν*}                                  // inhibit winner node
    G ← (1 − α)G                                      // decrease the gain
    i ← i + 1                                         // increment the adaptation step counter
until error < δ or i ≥ i_max
```

The original termination condition is based only on the maximal distance of a winner node to the city that is less than a given δ. However, in the case of poor convergence, e.g., due to the used approximation, the adaptation procedure is terminated after a given number of adaptation steps i_{max}. The city tour can be reconstructed from the ring of nodes because each city has a distinct winner. The used value of acceptable error is $\delta = 0.001$, and the used maximal number of steps is $i_{max} = 180$.

4.2. Co-adaptive net

The Co-adaptive net algorithm [11] also uses a randomization of the presented cities, but it does not use the inhibition mechanism. Instead, the winning number w_i is maintained for each neuron during the adaptation step. The required computational time of the select winner procedure is decreased by considering the restricted set of neighboring nodes of the previous winners. To avoid degenerate solutions, after every K adaptation steps, the winner is selected from the whole set of nodes.

One of the two adaptation procedures is selected according to the value of the gain G. In the case of $G < G_{cross}$, the winner node ν^* and its neighboring nodes are moved toward the city if and only if $w_\star = 0$. For $G \geq G_{cross}$, the winner and the neighbors (for $w_\star = 0$), or only the neighbors (for $w_\star = 1$) are moved; otherwise, none of nodes is adapted. The neighboring function is similar to the one used in the SME algorithm, but a node-specific gain is used $g_i = G(1 - |\nu_i, c|/\sqrt{2})$. The gain G is changed after each adaptation step by $G \leftarrow (1 - \alpha)G$ for $G \leq G_{cross}/2$, and $G \leftarrow (1 - 2\alpha)G$ otherwise.

Another important part of the Co-adaptive net is a construction of the city tour because the inhibition is not used. If a tour constructed from the winner nodes with $w_i = 1$ contains at least $\min\{n - 100, 0.98n\}$ cities after an adaptation step, winner nodes are found for the cities not in the tour considering the inhibition. The city tour may be constructed after each adaptation step, and the best-found tour (the shortest one) is returned as the solution of the TSP.

The adaptation is terminated if the winner nodes are sufficiently close to the cities (similar to the SME approach), if the neurons are in the same positions as they were at the end of the previous adaptation step, or if the current gain is small ($G \leq 0.01$), which is equivalent to the maximal number of adaptation steps.

Based on the results presented in [11] the following parameters are used as default settings of the Co-adaptive net in this paper: $m = 2.5n$, $G_0 = n/3$, $G_{cross} = 10$, $\alpha = 0.02$, $\mu = 0.626$, the size of the restricted set of nodes in the `select winner` procedure is $C^\star = 250$, the full search is performed after every 10 adaptation steps, and the size of the neighborhood in the `adapt` procedure is set to $\min\{2G + 1, 200, m/2\}$.

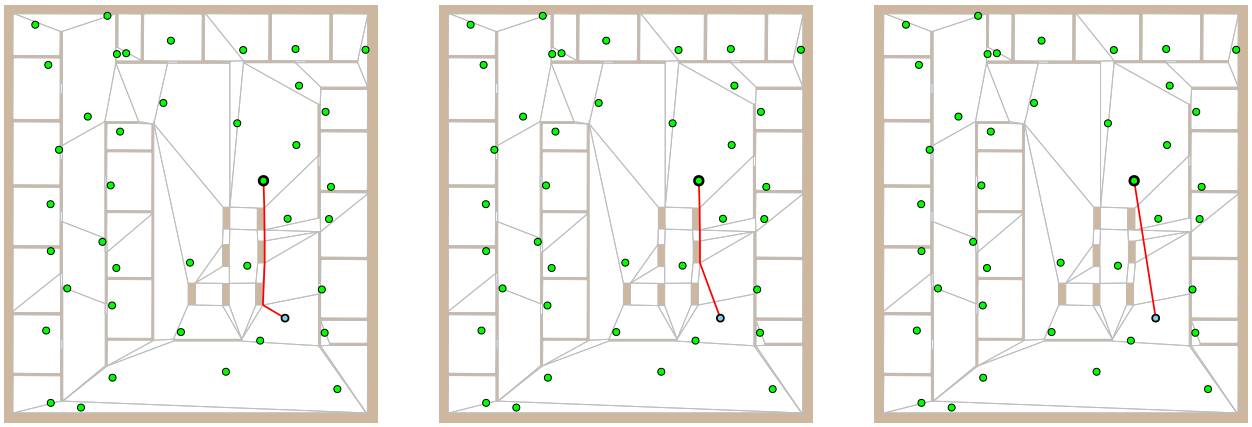
The authors of the Co-adaptive net use the center of the cities as the point around which a ring is initialized. Such a point can lie in an obstacle for the TSP in $\mathcal{W}$; therefore, alternative initializations are considered.

4.3. Approximation of the Shortest Path in the Polygonal Domain

A simple approximation of the shortest path based on a convex partition of the polygonal domain $\mathcal{W}$ has been used in [18]. The approximation is based on a refinement of a primary path found in a convex partition of $\mathcal{W}$. A convex partition $\boldsymbol{P}$ is a set of convex cells C_i, $\boldsymbol{P} = \{C_1, C_2, \ldots, C_k\}$ such that the union of the cells is $\mathcal{W}$, $\bigcup_{i=1}^{k} C_i = \mathcal{W}$. Cells are induced by the diagonals of $\mathcal{W}$, and each cell is formed from a sequence of $\mathcal{W}$ vertices. A node v is inside $\mathcal{W}$ during the adaptation; thus, it is always inside some cell C_v. The initial approximate path from v to the city c is found as the shortest path $S(w, c)$ over vertex w of C_v to c such that $w = \text{argmin}_{w_i \in C_v} |v, w_i| + |S(w_i, c)|$, where $|.,.|$ denotes the Euclidean distance between two points, and $|S(.,.)|$ is the length of the shortest path between two vertices, or vertex and city, see Fig. 1a.

The problem of finding the cell C_v is a point-location problem, which can be solved in $O(\log v)$ or in the average complexity $O(1)$ by the "bucketing" technique [14]. Alternatively, the cell can be determined during the node movement toward the city by the walking technique similar to [13]. The complexity of such cell determination is bounded by $O(\log n_d)$, where n_d is the number of passed diagonals of the used convex polygon partition.

The initial path can be improved by the following refinement procedure. Assume a node v inside the cell C_v and the approximation of the path from v to the vertex v_k as a sequence of vertices $(v_0, v_1, \ldots, v_k)$, $v_0 \in C_v$. A refinement is an examination of a direct visibility test between v and v_i. The visibility test is similar to [23], a convex partition is used instead of a triangulation. If a straight line from v to the vertex v_k crosses only diagonals or lies entirely in the same cell, then the vertex v_k is visible, and all vertices v_i for $i < k$ can be removed from the sequence. Examples of a refined path are shown in Fig. 1.

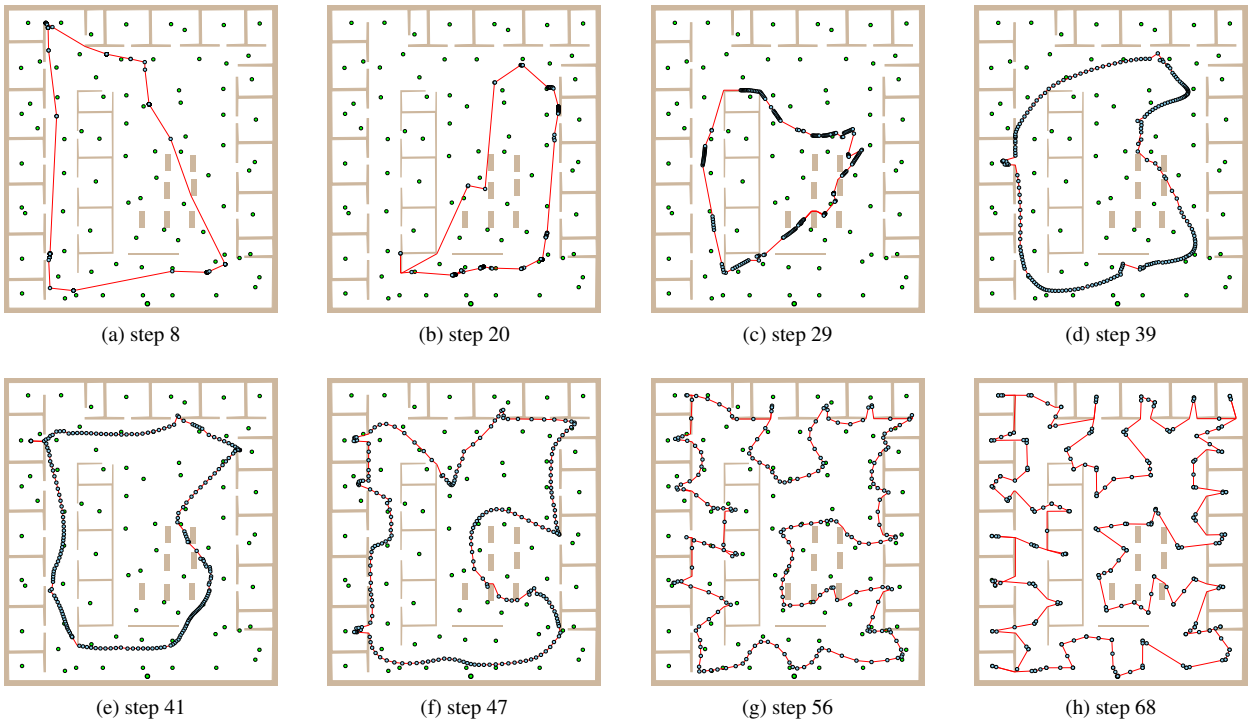


(a) path over v_0 (b) path over v_1 (c) full refinement

Figure 1: An example of path refinement, the gray segments represent diagonals of the convex partition, small disks are cities, and a node is connected with the city by the approximation of the shortest path (red segments).

The shortest paths from vertices to cities can be pre-computed by Dijkstra's algorithm in $O(ne \log(N_V + n))$, where n is the number of cities, N_V is the number of vertices, and e is the number of visible pairs (city-city, city-vertex, and vertex-vertex) of the complete visibility graph. The number of edges are bound by $e \leq N_V + N_V n$. The graph can be found in $O((N_V + n)^2)$ by the algorithm [29].

The adaptation process using the approximate path is visualized in Fig. 2. The nodes are connected by the approximate shortest paths between two nodes, which uses the same principle as the node–city paths, the vertices of the nodes' cells are considered. The path between nodes is not needed in the adaptation process; it is used only for the visualization.



(a) step 8 (b) step 20 (c) step 29 (d) step 39

(e) step 41 (f) step 47 (g) step 56 (h) step 68

Figure 2: An example of ring evolution in the environment *jh*; the small green disks represent cities, and the blue disks are nodes.

5. Modifications Used and Proposed

5.1. Approximation of the Shortest Path

Three variants of the refinement procedure of the approximate shortest path described in Section 4.3 have been considered in the experimental evaluation of the modified SOM algorithms. The refinement using only one vertex of the primary path over the vertex of the node cell is denoted as the *va-1* variant. Two additional variants are *va-0*, which does not use the refinement procedure, and *pa*, which represents the complete refinement of all vertices on the primary path.

Based on the results presented in [18], the *va-1* variant provides the best trade-off between the quality of the solutions and the required computational time. The *va-0* variant is faster, but the network does not converge in some cases due to imprecise approximations.

5.2. Select Winner Procedure

A path among obstacles in $\mathcal{W}$ has to be found to determine the winner node of the current presented city to the network, which means m node–city distance queries have to be performed for each presented city. However, the required computational time can be reduced if the Euclidean distance of the node to the city is considered before the node–city distance is queried. If the Euclidean node–city distance is longer than the Euclidean distance of the current winner candidate, it is not necessary to determine the path among obstacles. This Euclidean pre-selection is denoted as the *euclid-pre* select winner method in the experimental part of this paper.

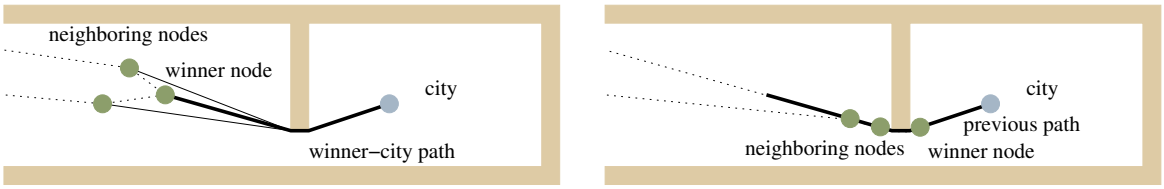
Moreover, after several adaptation steps, the winners are preserved over the steps. Thus, the previous winner to the city can be used as the initial winning candidate. Such an initial selection of the winner candidate can avoid unnecessary computations of the shortest path. In the final adaptation steps, winners are very close to cities, and a city and its winner node are typically in the same cell; in other cases, the shortest path can be just a straight line segment. Therefore, the determination of node–city distance can be very fast, and the Euclidean distance is sufficient to confirm

that the previous winner is really the closest node to the city. This winner selection method with the Euclidean distance pre-selection is denoted as *informed*.

These improvements can be considered technical, because they do not affect the quality of the solution found and only decrease the required computational time at the cost of a more complex algorithm.

5.3. Adaptation Rule

The `adapt` procedure is more complex than the `select winner` procedure because a path has to be retrieved in the node–city path query and the adapted node is moved towards the city, i.e., a particular straight line segment of the path has to be determined. The node v is moved closer to the city c proportionally to the node–city distance $D(v, c)$, learning rate, and neighboring function. The distance of v to c is decreased about $\beta D(v, c)$, where β has the form $\beta = \mu exp(-l^2/G^2)$. The value of β decreases with the increasing distance of the neighboring node. It also decreases with each adaptation step, as the learning gain G decreases. If β is very small, the movement can be negligible; therefore, once the β is under a given threshold, the adaptation of neighboring nodes can be omitted. This modification of the adaptation rule is called $\beta - condition$ in this paper, and it can be used for rules without decreasing the neighborhood size. The influence of this modification has been experimentally examined for the SME and Co-adaptive net algorithms.



(a) before adaptation (b) after adaptation

Figure 3: Utilization of the winner–city path for the neighboring nodes.

An additional speed improvement of the `adapt` procedure can be based on the usage of the winner path to the city c for the neighboring nodes. If nodes are close to each other, and if a path contains a map vertex (avoiding an obstacle), a path from the neighboring nodes will likely pass the same vertex. Thus, the neighboring node v can be moved along the same path as the winner node $v^\star$, while the distance is decreased by the Euclidean distance between $v^\star$ and v, i.e., v is placed at the position of $v^\star$ before its movement and adaptation toward c. The situation is schematically shown in Fig. 3. This modification of the adaptation rule is called *approx. adapt*, and it is combined with the $\beta - condition$ modification.

5.4. Adaptation Parameters

Beside the original adaptation parameters of the SME and Co-adaptive net algorithms, the following modifications are considered as well. The rules proposed in [39] and denoted as *Zhang-Bai-Hu* rules are also used in SME. Furthermore, the original SME adaptation rule is complemented by the decreasing size of the winner node neighborhood, i.e., the size d is updated to $d = 0.98d$ at the end of each adaptation step.

The multiple scale neighborhood functions used by Murakoshi and Sato in [28] are utilized in the Co-adaptive net; this modification is denoted by the abbreviation MSNF for short.

5.5. Initialization

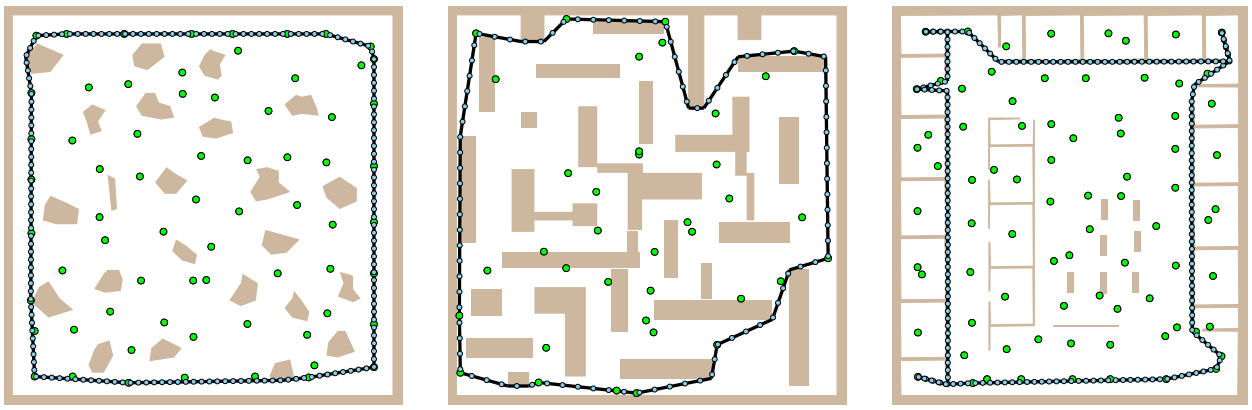
Due to obstacles, the initialization of the nodes used for the Euclidean TSP described in [5] cannot be directly used in the polygonal domain $\mathcal{W}$. That is why the following initializations are considered in the experimental examination of the modified algorithms.

The first initialization method is called *first* because the first city is used to initialize the ring of nodes as a circle with a small radius (5 mm) around the city. The small radius ensures that the nodes are placed in $\mathcal{W}$, as cities are always placed at a greater distance from the obstacles.

The second method uses the closest city to the centroid of cities, and a ring is also created as a small circle with the same radius like in the first method. The method is called *center* in this paper.

The third initialization is similar to the *center* method, but the center of the circle is selected as the city with the smallest standard deviation of the distances to other cities. The method is called *dev*.

Inspired by approach [7], the last examined initialization method is called *hull* because it is based on the convex hull of the cities. The cities at the border of the convex hull are connected by the shortest paths. The connected cycle is then used to initialize nodes equidistantly along the cycle. Examples of the *hull* initialization are shown in Fig. 4.



(a) *potholes* (b) *warehouse* (c) *jh*

Figure 4: Examples of the *hull* initialization, the green disks are cities, small blue disks are nodes, and the bold black line segments represent a connected ring of nodes.

6. Performance Evaluation

The performance of the SOM algorithms is evaluated for a set of inspection planning problems[1]. The environments are represented by polygonal maps. The name of the environment with a subscript denoting the visibility range ρ in meters represents the particular TSP. It means that a robot performing measurement at the city position senses its surrounding environment in the distance ρ [17]. Parameters of the environments are shown in Table 1, where N_V is the number of vertices, N_H is the number of holes, and N_C is the number of convex cells of the supporting convex partition. Environments *jh*, *pb*, *ta*, and *h2* represent maps of real buildings; thus, they provide a representative problem in size. In particular, maps *jh*, *pb*, and *ta* have been used as experimental sites for search and rescue scenarios in the PeLoTe project [24].

The algorithms have been implemented in C++ and compiled by the G++ 4.2 with the -O2 optimization flag. All results have been obtained within the same computational environment using single core of the Athlon X2 CPU running at 2 GHz, 1 GB RAM, and FreeBSD 8.1. Thus, all required computational times presented can be directly compared.

The cities are applied to the network in a random order in all examined algorithms, and therefore, each particular algorithm variant is executed twenty times for each problem, and average values are determined. The quality of solutions is evaluated as the percentage deviation to the optimum tour length of the mean solution value, $PDM = (\overline{L} - L_{opt})/L_{opt} \cdot 100\%$, and as the percentage deviation from the optimum of the best solution value (PDB), where L_{opt} is the length of the optimal solution found by the Concorde solver [2]. The PDM and PDB have variances due to randomization, therefore a tolerance between a half and one percent is considered in the quality evaluation of solutions found by the particular modified algorithm.

[1]The problems with necessary supporting structures are available at `http://purl.org/faigl/tsp/`.

Table 1: Testing environments with obstacles

Name	**Dimensions** [m × m]	**Area** [m^2]	N_V	N_H	N_C
jari	4.5 × 4.9	20	48	1	14
complex2	20.0 × 20.0	322	40	3	21
m1	4.8 × 4.8	20	51	4	26
m2	4.8 × 4.8	15	51	6	20
map	4.8 × 4.8	14	68	8	36
potholes	20.0 × 20.0	367	153	23	75
rooms	20.0 × 20.0	351	80	0	33
a	8.9 × 14.1	71	99	6	22
dense	21.0 × 21.5	299	288	32	150
m3	4.8 × 4.8	17	308	50	120
warehouse	40.0 × 40.0	1192	142	24	83
jh	20.6 × 23.2	455	196	9	77
pb	133.3 × 104.8	1453	89	3	41
ta	39.6 × 46.8	731	74	2	30
h2	84.9 × 49.7	2816	2062	34	476

The speed improvement of a particular algorithm variant is measured as the ratio of the average required computational times of the original algorithm and its modified variant. The required computational time consists of the preparation time T_{init} and the time needed to adapt the network T_{adapt}. The preparation phase is a creation of supporting structures: the convex polygon partition, visibility graph, and shortest paths between cities and map vertices. The convex partition is found in tens of millisecond using Seidel's algorithm [33], and the construction of the complete visibility graph takes 41 millisecond for the largest problem $h2_2$ with 575 cities and 2062 map vertices. These times are negligible according to the total required computational time, and they are not included in the presented time values. The most time consuming preparation part is determination of the shortest path between cities and vertices. This time is included in the presented total required computational time denoted as T. Regarding the preparation time the speed improvement of a particular algorithm variant is computed from T_{adapt}.

The adaptation procedure itself is composed of the selection of winners and adaptation toward cities. The particular required computational times in these parts are useful for determining the most computationally intensive part of the algorithm. Therefore, $\%T_s$ and $\%T_a$ denote computational times spent in the particular part of the adaptation procedure (`select winner` and `adapt` respectively) in percentages of the total adaptation time T_{adapt}.

To avoid presentation of many detailed results, the examined problems are organized into three sets according to the number of cities, see Table 2. In the overall comparison of the examined algorithms' modifications, T_{adapt} for the original algorithm is the reference computational time, i.e., an average computational time for each problem of the reference algorithm is divided by the average T_{adapt} for the algorithm variant. The speed improvement, denoted as *Sp.*, is computed as an average value of improvements over all problems in the set.

6.1. The SME Algorithm

The original Somhom's adaptation procedure has been augmented by the algorithm to find the approximate shortest path in $\mathcal{W}$. The *pa* refinement variant and the pure *geodesic* winner selection are used. Besides, the tour length at each adaptation step is computed, and the best tour found during the adaptation is used as the found solution. This algorithm variant is used as the reference algorithm in the presented experimental results of SME algorithm modifications. This variant is used as the base algorithm for other examined modifications as follows.

Table 2: Problems sets

Small set		***Middle set***		***Large set***	
problem name	*n*	*problem name*	*n*	*problem name*	*n*
jari	6	$dense_4$	53	$potholes_1$	282
complex2	8	$potholes_2$	68	jh_1	356
m1	13	$m3_1$	71	$pb_{1.5}$	415
m2	14	$warehouse_4$	79	$h2_2$	568
map	17	jh_2	80	ta_1	574
potholes	17	pb_4	104		
rooms	22	ta_2	141		
a	22	$h2_5$	168		

Table 3: The SME algorithm - improvements of the `select winner` procedure with the *va-1* refinement

Problems	***select winner - geodesic***				***select winner - euclid-pre***				***select winner - informed***			
	PDM	**PDB**	**Sp.**	**Steps**	**PDM**	**PDB**	**Sp.**	**Steps**	**PDM**	**PDB**	**Sp.**	**Steps**
small	1.71	0.00	1.2	64	1.10	0.01	2.0	64	1.36	0.01	2.0	64
middle	4.94	2.18	1.3	84	4.77	2.00	2.1	84	4.61	2.38	2.1	84
large	3.94	2.99	1.3	100	4.16	3.04	2.2	100	4.00	3.05	2.1	100

First, the select winner methods described in Section 5.2 have been considered with the *va-1* refinement variant, the results are presented in Table 3. The *Sp.* column shows how many times the performance of the algorithm has been improved in comparison to the reference algorithm with the *pa* refinement and the pure *geodesic* winner selection. In the case of the *geodesic* winner selection, the algorithm spent about forty five percentage points in the select winner procedure, and about fifty five percentage points in the `adapt` procedure. After applying the Euclidean pre-selection of winner node candidates, the dominant algorithm part is the `adapt` procedure. Consideration of the previous winner does not significantly reduce the required computational time, and the results are pretty much similar to the *euclid-pre* variant. The PDM variances of the select winner methods are below 0.5 % threshold; thus, the overall quality of solutions is considered to be same.

The most time consuming part of the SME algorithm with the *informed* `select winner` procedure is the `adapt` procedure, therefore, modifications of the procedure have been examined. The experimental results with modified adaptation rules described in Section 5.3 are presented in Table 4. The *informed* select winner method and the *va-1*

Table 4: The SME algorithm - adaptation rule modifications

Problems	***original adaptation rule***				***β-condition modification***				***approx. adapt. + β-condition***			
	PDM	**PDB**	**Sp.**	**Steps**	**PDM**	**PDB**	**Sp.**	**Steps**	**PDM**	**PDB**	**Sp.**	**Steps**
small	1.36	0.01	2.0	64	1.29	0.01	2.4	64	1.62	0.01	2.4	64
middle	4.61	2.38	2.1	84	4.58	2.46	3.3	84	5.23	2.86	3.2	84
large	4.00	3.05	2.1	100	4.03	2.87	4.0	100	4.65	3.63	3.9	100

refinement are used, and the $\beta - condition$ is set to $\beta = 10^{-5}$. The $\beta - condition$ effectively decreases the active neighborhood of the winner node, which decreases the required computational time without noticeable degradation of

the solution quality. The *approx. adapt* modification does not provide any improvements, and the solution quality is also worse. In all cases, solutions are found in the same average number of the adaptation steps, see the column *Steps*.

Table 5: Influence of the adaptation parameters to SME with the *va-1*, *informed*, and β − *condition* modifications

Problems	*original adapt. param.*			*orig. with decreasing* d				*Zhang-Bai-Hu rules*			
	PDM	**Sp.**	**Steps**	**PDM**	**PDB**	**Sp.**	**Steps**	**PDM**	**PDB**	**Sp.**	**Steps**
small	1.29	2.4	64	4.53	0.36	8.1	163	5.71	0.13	9.1	160
middle	4.58	3.3	84	8.16	4.36	7.6	168	6.15	2.52	21.3	71
large	4.03	4.0	100	3.99	3.14	6.5	100	9.21	4.17	86.4	49

Additional speed improvement is achieved using modifications of adaptation parameters described in Section 5.4. The experimental results are shown in Table 5. For the *small* and *middle* sized problems decreasing the size of the winner node neighborhood leads to an algorithm three times faster, but the solution quality is worse and a higher number of adaptation steps is needed. The results indicate that for these problems the restriction of the neighborhood is too strong, mainly at the beginning of the adaptation. However, for *large* problems the initial number of nodes seems to be sufficiently high (possibly unnecessarily high), as the solution quality is preserved. The Zhang-Bai-Hu rules dramatically reduces the required computational time for problems with a higher number of cities, although the solution quality is more than two times worse for larger problems.

Table 6: SME with the *va-1* refinement, *informed*, β − *condition* modifications, Zhang-Bai-Hu rules, and *hull* initialization

Problems	*va-0 refinement*			*va-1 refinement*				*pa refinement*			
	PDM	**Sp.**	**Steps**	**PDM**	**PDB**	**Sp.**	**Steps**	**PDM**	**PDB**	**Sp.**	**Steps**
small	6.03	9.8	177	5.62	0.08	8.3	166	5.96	0.16	8.3	160
middle	6.93	30.8	89	5.00	2.60	22.2	63	4.80	2.86	20.7	65
large	12.55	116.5	64	4.44	3.51	105.7	43	4.33	3.02	99.4	43

The poor solution quality of the Zhang-Bai-Hu rules is improved by the *hull* initialization, see results for the path refinement variants presented in Table 6. The most significant improvement in the solution quality and also in the required computational time is for large problems. Other initialization methods do not increase the solution quality, which is also the case for other examined modifications of the SME algorithm. The significant reduction of the number of the shortest path queries allows consideration of the full path refinement variant, although the benefit is not evident from the presented results. The algorithm with the modifications used has the `select winner` and `adapt` parts almost equally computationally intensive regarding $\%T_s = 46$ % and $\%T_a = 49$ % for large problems.

The additional speed improvements can be achieved by a more restricted size of the winner node neighborhood. However, using initial size $m/8$ only decreased the solution quality without significant speed improvement. The multiple scale neighborhood functions do not improve the solution quality; thus, these results are not presented.

6.2. The Co-adaptive Net Algorithm

The *informed* `select winner` procedure with the *pa* refinement is used in the Co-adaptive net algorithm. Even though authors of the Co-adaptive net algorithm initialized the ring as a small circle around cities' centroid, the four initialization methods described in Section 5.5 do not provide significant differences in the solution quality nor the computational requirements, and therefore, the *first* initialization method is used as default. This algorithm variant is the reference algorithm (of the required computational time) in the overall comparisons of its examined modifications.

Similarly to the SME algorithm, the *va-1* refinement does not provide noticeable changes in the solution quality in comparison with the full path refinement; however, the performance is improved by about more than ten percentage

Table 7: Influence of adaptation rule modifications to the Co-adaptive net with the *va-1* refinement and *informed* select winner procedure

Problems	*original adaptation rule*				*β-condition*				*β-condition + MSNF*			
	PDM	**PDB**	**Sp.**	**Steps**	**PDM**	**PDB**	**Sp.**	**Steps**	**PDM**	**PDB**	**Sp.**	**Steps**
small	1.49	0.46	1.1	174	1.33	0.11	1.1	175	1.22	0.07	1.1	196
middle	6.28	3.05	1.2	290	6.59	3.20	1.2	290	4.56	2.13	1.0	311
large	6.60	4.89	1.2	388	6.64	4.87	1.2	389	5.39	4.10	1.0	390

points. The original adaptation rule is modified to consider the $\beta - condition$, then the rule is combined with the Multi Scale Neighborhood Functions (MSNF). The results for the three problems sets are presented in Table 7. Notice, the Co-adaptive net requires a higher number of adaptation steps than the SME algorithm. However, the total required computational time is lower because less nodes are involved in the `select winner` and `adapt` procedures. Considering MSNF increases the solution quality and the number of required adaptation steps. Here, it should be noted that the network adaptation has been terminated by the $G < 0.01$ condition in all algorithm variants. The minimal distance of the winner node to the city is significantly higher than in the SME algorithm, i.e., by units or tens in comparison to Somhom's $\delta = 0.001$.

Table 8: An influence of the gain-decreasing rate α to the Co-adaptive net with the *va-1* refinement and *informed* select winner procedure

Problems	*original adaptation rule*						*β-condition + MSNF*					
	$\alpha = 0.05$		$\alpha = 0.1$		$\alpha = 0.2$		$\alpha = 0.05$		$\alpha = 0.1$		$\alpha = 0.2$	
	PDM	**Sp.**	**PDM**	**Sp.**	**PDM**	**Sp.**	**PDM**	**Sp.**	**PDM**	**Sp.**	**PDM**	**Sp.**
small	2.59	2.6	3.49	4.8	4.31	8.8	2.03	2.6	2.59	4.6	4.47	8.6
middle	7.93	2.9	8.36	5.6	10.10	11.4	5.30	2.6	6.58	5.1	7.87	10.0
large	7.43	3.0	10.27	5.8	23.05	12.4	6.08	2.5	6.94	5.1	8.87	10.4

The used gain-decreasing rate $\alpha = 0.02$ is relatively small, and the computational burden can be decreased by a higher value. The experimental results for various α are presented in Table 8. The original Co-adaptive net algorithm is very sensitive to changes of α while MSNF provides significantly better results. The value $\alpha = 0.1$ provides almost the same solution quality (about one or two percent worse) as the original algorithm, and it is more than four times faster. Also in this case, another initializations of the ring do not provide any significant improvements.

6.3. Algorithms Comparison

Based on the examination of described modifications two new variants of the SME and the Co-adaptive net algorithms are selected as successors of their originals. The applied modifications are selected as the best trade-off between the solution quality and required computational time, mainly concerning the $h2_2$ problem. Particular parts of the original algorithms and the proposed modifications are as follows.

The full path refinement *pa* and the pure *geodesic* variant of the `select winner` procedure are used in the original SME algorithm. The nodes are initialized around the first city. The *pa* refinement is also used in the proposed successor of the SME algorithm, as the computational burden is only slightly increased in comparison with the *va-1* variant. The *informed* `select winner` procedure, the *hull* initialization method, $\beta - condition$, and the Zhang-Bai-Hu adaptation rules are utilized. In both Co-adaptive net algorithms, the *informed* modification of the `select winner` procedure with the *pa* path refinement are utilized. The initial size of the winner neighborhood is set to $m/2$. In the case of the original Co-adaptive net, the parameters described in Section 4.2 are used, and nodes are initialized by the method *dev*, which provides the highest solution quality for large problems. Nodes are initialized by the *center*

Table 9: Proposed modifications of the original algorithms

A Part of the Adaptation Procedure	**Modified SME**	**Modified Co-adaptive Net**
Initialization	*hull*	*center*
Select Winner	*informed*	*informed*
Adaptation Rule	$\beta - condition$	$\beta - condition$
Neighbourhood Function(s)	-	MSNF
Adaptation Parameters/Rule	Zhang-Bai-Hu	$\alpha = 0.1$

initialization method in the modified Co-adaptive net algorithm that uses the $\beta - condition$ with MSNF. The only changed parameter is the gain-decreasing rate $\alpha = 0.1$, which decreases the computational burden without significant solution quality changes. The proposed modifications of the particular part of the adaptation procedures are depicted in Table 9, where '-' denotes the original part the algorithm.

Detailed performance results of the original and the modified algorithms are presented in Table 10. To provide an overview of the algorithms' performance, average values of the required computational time and the solution quality measured by the PDM are shown in Fig. 5 as histograms of the problem size. Selected solutions found by the modified SME algorithm are presented in Fig. 6.

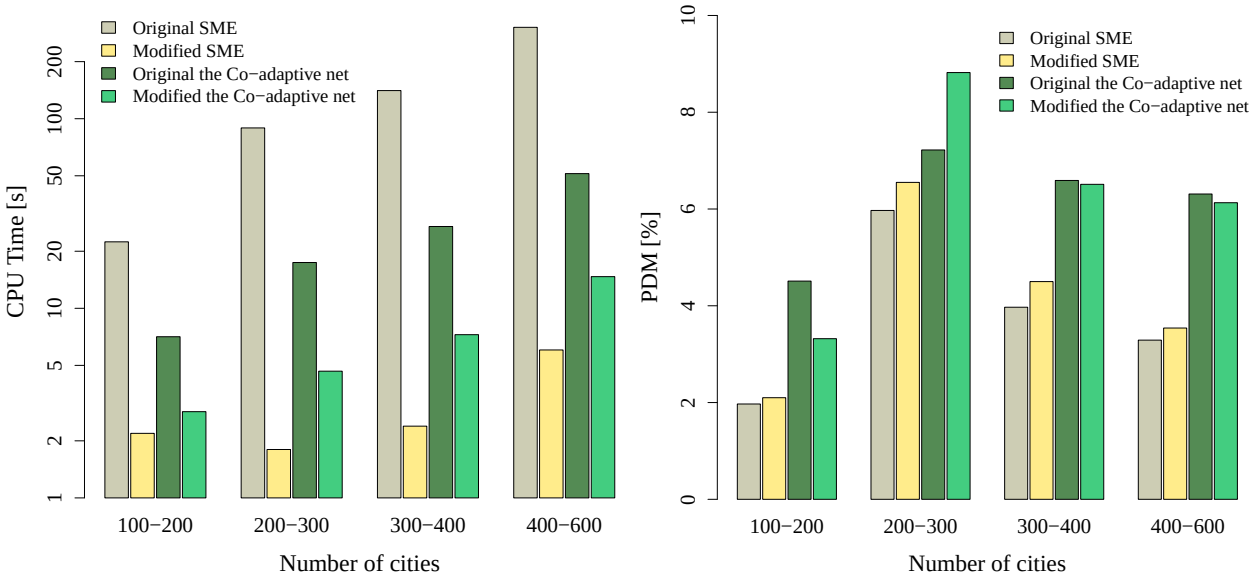
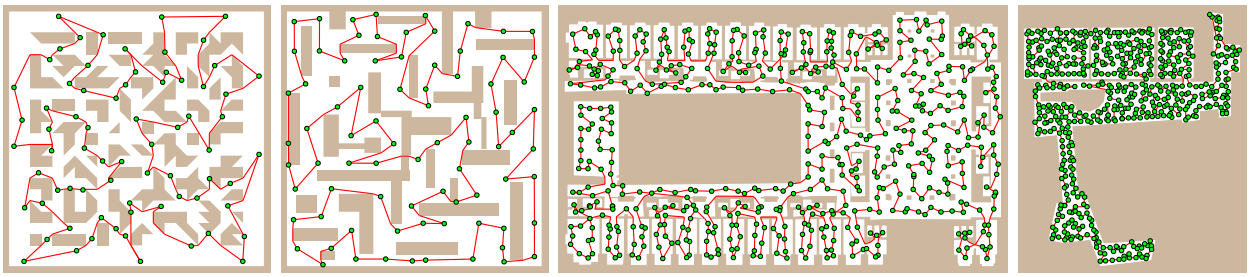


Figure 5: Average values of the required computational time and the solution quality.

Regarding the presented results the modified SME algorithm provides superior results for *middle* and *large* problem sets. For problems with less than fifty cities the modified Co-adaptive net algorithm provides better results. This can be caused by a fewer number of neighboring nodes used in the SME algorithm in comparison to the Co-adaptive net algorithm. Besides, the Co-adaptive net uses the winning number, and the algorithm avoids adaptation of the winners and neighboring nodes, which means the nodes are moved with less frequency than in the SME algorithm. The performance of the Co-adaptive net algorithm in the examined large non-Euclidean TSP is quite surprising. Even though several modifications and parameter settings have been used, the algorithm does not provide competitive results to the modified SME algorithm. The original SME algorithm provides solutions with significantly higher quality, which is not the case of the TSPLIB problems presented in [11]. The applied modifications to Somhom's algorithm significantly decrease the required computational time, and make *penalization* by the shortest path determination less



(a) $m3_1$, L=40.9 m (b) $warehouse_4$, L=381.3 m (c) $h2_2$, L=1339.1 m (d) ta_1, L=567.8 m

Figure 6: Selected solutions found by the modified SME algorithm, the small green disks represent cities that are connected by the shortest path among obstacles using the complete visibility graph.

important. From a certain point of view, the modified algorithm in the non-Euclidean problem starts to be competitive to the original algorithm in the Euclidean problem, e.g., according to results presented in [15].

Table 10: The algorithms performance

Name	n	L_{opt} [m]	***The SME algorithm***						***The Co-adaptive Net***					
			original			*modified*			*original*			*modified*		
			PDM	**PDB**	**T** [s]	**PDM**	**PDB**	**T** [s]	**PDM**	**PDB**	**T** [s]	**PDM**	**PDB**	**T** [s]
jari	6	13.6	0.00	0.00	0.02	1.06	0.00	0.01	0.00	0.00	0.03	0.10	0.00	0.01
complex2	8	58.5	0.00	0.00	0.04	8.77	0.00	0.01	0.00	0.00	0.05	1.30	0.00	0.01
m1	13	17.1	0.00	0.00	0.09	4.64	0.00	0.02	0.05	0.00	0.07	0.63	0.00	0.02
m2	14	19.4	8.64	5.32	0.10	13.12	0.00	0.02	2.25	0.00	0.08	6.97	0.00	0.03
map	17	26.5	1.98	0.00	0.17	10.31	0.00	0.04	3.22	0.00	0.14	3.81	0.00	0.05
potholes	17	88.5	1.11	0.00	0.31	3.65	0.00	0.10	0.94	0.00	0.24	1.99	0.00	0.11
a	22	52.7	0.01	0.00	0.38	1.95	0.00	0.06	0.30	0.00	0.22	0.79	0.00	0.08
rooms	22	165.9	0.60	0.08	0.36	4.18	1.27	0.06	3.47	1.02	0.23	4.14	2.26	0.06
dense_4	53	179.1	15.40	9.12	2.95	12.14	7.28	0.53	10.22	4.76	1.29	12.48	8.08	0.51
potholes_2	68	154.5	5.21	3.37	4.51	5.55	3.54	0.42	4.97	1.54	1.39	7.08	4.19	0.42
m3_1	71	39.0	6.23	4.25	5.05	6.88	4.72	0.69	8.89	3.63	1.85	8.86	5.54	0.69
warehouse_4	79	369.2	5.26	2.19	5.54	5.69	3.27	0.49	7.16	2.76	1.66	7.23	2.60	0.48
jh_2	80	201.9	1.63	0.28	6.26	1.82	0.43	0.51	5.50	3.58	1.80	3.68	2.26	0.54
pb_4	104	654.6	1.00	0.02	7.24	0.70	0.04	0.37	0.84	0.15	2.26	1.58	0.10	0.54
ta_2	141	328.0	3.22	2.33	13.44	3.33	2.43	0.46	5.60	3.73	3.69	4.31	2.44	0.86
h2_5	168	943.0	1.70	0.91	46.58	2.26	1.17	5.73	7.10	4.54	15.29	4.08	2.55	7.14
potholes_1	282	277.3	5.97	3.93	89.42	6.55	4.00	1.80	7.22	4.89	17.44	8.82	7.25	4.66
jh_1	356	363.7	3.97	3.03	140.75	4.50	3.06	2.39	6.59	3.59	27.01	6.51	4.49	7.26
$\text{pb}_{1.5}$	415	839.6	2.10	1.32	133.99	1.84	1.38	2.07	2.96	1.68	24.97	3.31	2.34	6.07
h2_2	568	1316.2	2.29	1.20	516.90	2.79	1.74	12.35	9.11	7.30	89.89	6.41	4.32	28.29
ta_1	574	541.1	5.48	4.24	259.38	5.99	4.93	3.68	6.85	4.72	39.14	8.65	7.21	9.70

6.4. Modified SME Algorithm Discussion

A detailed insight into the performance results of the modified SME algorithm gives several interesting observations, as is shown in Table 11. The worst performance of the algorithm in the *small* problems is related to the poor convergence as can be seen from the number of required adaptation steps. The presented results are average values over twenty runs; thus, the problems with 180 steps do not converge at all. The reason for this may be an excessively small value of the learning gain G together with the decreasing size of the neighborhood. However, the final found route (the length is denoted as L_{best}) is found very early, in S_{best} steps. So, worse solutions are found in the consecutive steps. The tour found in the last step is about units of percentage points worse than the best found tour, which is indicated in the column PDM_{last}. Although this is not the expected behaviour, the computational requirements are lower than for the original algorithm. These results indicate further potential improvements of the algorithm.

The minimal distance of winners to the cities is also affected by the poor convergence. Notice the *Error* values are in centimeters, due to default units of the maps used. Even though this error is not too important in the combinatorial TSP, as the solution can be considered as a sequence of cities, the error is crucial in other routing problems in the polygonal domain, e.g., the watchman [16] or safari route problem [15]. The difference is that in these problems, the ring may be the route itself, and not a representation of a route over cities. Here, it is worth mentioning that for the Co-adaptive net algorithm the error is negligible for *small* problems, and equals to tens of centimeters for larger problems.

The columns T, T_{init}, and T_{adapt} show the total required computational time, and particular times spent in the initialization and adaptation procedures. All shortest paths between cities and also from all map vertices to the cities are determined in the initialization. In several cases, T_{init} is similar to T_{adapt}. The initialization time is even greater than the adaptation time for the problem $h2_2$. The last column T_{LK} shows average values of the required computational time to find a solution by the `linkern` algorithm from the Concorde package [2], which uses the Chained Lin-Kernighan heuristic. The algorithm utilized a distance matrix that is found in time T_{init}; thus, the last two columns can be used to compare the computational burden of SOM and the combinatorial heuristic. In three cases, the SOM adaptation procedure is less computationally expensive than the heuristic approach. These results are particularly interesting because the used path approximation is a relatively complex algorithm in comparison with the usage of the distance matrix in the combinatorial heuristic.

Regarding T_{init} and T_{adapt} for the $h2_2$ problem, the most intensive part of the algorithm is the preparation of all the shortest paths. These paths are not involved in the adaptation process, and therefore, an additional speed improvement can be based on omitting the paths pre-computation, and consideration of approximate paths determined during the adaptation. The solution quality can decrease; so, the idea would need further investigation.

Table 11: Detail performance results of the modified SME algorithm

Name	n	L_{opt} **[m]**	L_{best} **[m]**	**Steps**	$\mathbf{S}_{best}$	**PDM**	$\mathbf{PDM}_{last}$	**Error [cm]**	**T [s]**	$\mathbf{T}_{init}$ **[s]**	$\mathbf{T}_{adapt}$ **[s]**	$\mathbf{T}_{LK}$ **[s]**
jari	6	13.6	13.70	97	1	1.06	2.87	5.7860	0.01	0.006	0.004	0.001
complex2	8	58.5	63.59	164	1	8.77	11.09	46.8591	0.01	0.004	0.008	0.001
m1	13	17.1	17.85	124	4	4.64	6.06	3.3118	0.02	0.007	0.009	0.001
m2	14	19.4	21.99	172	5	13.12	14.75	21.1784	0.02	0.007	0.014	0.001
map	17	26.5	29.26	180	6	10.31	11.79	18.0338	0.04	0.014	0.019	0.001
potholes	17	88.5	91.78	180	7	3.65	4.01	61.3482	0.10	0.074	0.019	0.001
a	22	52.7	53.77	180	9	1.95	2.67	39.7323	0.06	0.024	0.031	0.001
rooms	22	165.9	172.83	180	9	4.18	4.81	59.9355	0.06	0.021	0.032	0.091
dense_4	53	179.1	200.86	44	16	12.14	12.27	0.0008	0.53	0.286	0.235	0.018
potholes_2	68	154.5	163.13	42	16	5.55	5.58	0.0007	0.42	0.119	0.295	0.028
m3_1	71	39.0	41.74	47	16	6.88	7.05	0.0102	0.69	0.332	0.358	0.029
warehouse_4	79	369.2	390.18	139	16	5.69	5.76	1.4492	0.49	0.099	0.383	0.040
jh_2	80	201.9	205.61	42	16	1.82	1.87	0.0008	0.51	0.174	0.328	0.045
pb_4	104	654.6	659.18	45	18	0.70	0.73	0.0008	0.37	0.068	0.297	0.239
ta_2	141	328.0	338.94	43	17	3.33	3.35	0.0008	0.46	0.085	0.368	0.681
h2_5	168	943.0	964.24	119	19	2.26	2.31	0.9978	5.73	4.417	1.278	0.947
potholes_1	282	277.3	295.51	42	17	6.55	6.56	0.0008	1.80	0.633	1.144	0.233
jh_1	356	363.7	380.07	42	18	4.50	4.50	0.0008	2.39	0.837	1.526	0.867
$\text{pb}_{1.5}$	415	839.6	855.03	43	19	1.84	1.85	0.0008	2.07	0.585	1.458	1.978
h2_2	568	1316.2	1352.89	44	20	2.79	2.80	0.0008	12.35	8.004	4.280	4.396
ta_1	574	541.1	573.52	43	20	5.99	6.00	0.0008	3.68	1.434	2.201	0.935

6.5. Discussion

Two state-of-the-art SOM algorithms have been examined for the non-Euclidean TSP in the polygonal domain $\mathcal{W}$. The algorithms have been improved in several ways by already published modifications of the adaptation rule, and also by new proposed improvements. One of the main issues of SOM application in $\mathcal{W}$ is determination of node–city path, which is computed many times. The presented modifications significantly reduce the number of node–city path queries, and reduce the required computational time up to one hundred times.

The improvements are mostly visible for problems with a high number of cities. However, SOM approaches for the Euclidean TSP are able to solve problems with thousands of cities. In the presented results, the largest examined problem has "only" about five hundreds cities. From the practical point of view, the largest examined problems are the inspection planning problems within real environments and quite small visibility range[2]. The considered problems represent a realistic upper bound of the problem size because for a higher number of cities the visibility range has to be unrealistically small, or the environments have to be significantly larger. The visibility range and the size of the environment relate with a structure of cities (sensing locations in the inspection planning) in an environment, which can affect the solution quality. Cities that are relatively close to each other make the local search for a shorter route more important, which is a quite difficult task for a conventional SOM; mainly because of the decreasing learning rate during the adaptation. In these cases, the proposed *hull* initialization improves the quality because the adaptation starts with a spread ring. The modified algorithms have been examined only in $\mathcal{W}$ and it is expected that the benefit of the presented modifications will not be significant in the Euclidean problems due to relatively inexpensive computation of node–city distances.

An additional performance improvements can be based on consideration of smaller or dynamic number of nodes. In literature, 2.5 times more nodes than cities has been reported as the most suitable. Also for the examined problems and algorithms, different numbers of nodes decrease the solution quality. The idea of the *approx. adapt* modification does not provide expected improvement. However, in early adaptation steps, nodes are often moved over map vertices that mean the neighboring nodes of the winner node are placed at the same shortest path from the particular map vertex to the city. In these cases, new nodes can be created and adaptation can start with only a very small number of nodes, which is an idea for future work.

One remark about the Co-adaptive net algorithm and the proposed improvements has to be mentioned. The algorithm is quite complex, which can be considered as a weak point of an eventual massive parallelization. This is also a weak point of the determination of the geodesic path, and the applied improvements to the `select winner` procedure, which can increase the complexity of a parallel implementation. Thus, it seems that one of the SOM features is lost in $\mathcal{W}$.

Another point of the Co-adaptive net algorithm is its relatively strict orientation to the routing optimization, which, in fact, is not an issue for the TSP. The modified SME algorithm provides much better performance in this aspect, i.e., the maximal distance of the winner nodes to cities. From this perspective, the modified Somhom's adaptation schema with the Zhang-Bai-Hu adaptation rules is more suitable for other routing problems in $\mathcal{W}$. Consideration of the ring evolution in $\mathcal{W}$ provides opportunity to find a solution of the watchman route problem [16] or other inspection problems where cities are not explicitly prescribed, which is one of the main SOM benefits over the classical combinatorial approaches [15].

7. Conclusion

Improved self-organizing map-based algorithms for the TSP in the polygonal domain have been proposed. The required computational time of the algorithms has been decreased by the proposed β − *condition* adaptation rule and the *informed* select winner procedure in combination with the approximate shortest path in $\mathcal{W}$. In addition, the performance of the SME algorithm has been improved using a combination of the Zhang-Bai-Hu adaptation rules with the new *hull* initialization technique. The successor of the Co-adaptive net algorithm utilizes the MSNF adaptive rule with the *center* initialization to improve the quality of solutions.

The algorithms have been examined in several instances of the inspection planning task in the polygonal domain $\mathcal{W}$. The proposed algorithms move the performance of the SOM algorithms in $\mathcal{W}$ to the next level, and allow

[2]The visibility range in meters is denoted as the subscript of the problem name.

their further application in other routing problems in the polygonal domain. The complexity of non-Euclidean distance determination is indicated in the SOM literature as a drawback. The encouraging results presented in this paper, and the significant performance improvements can be motivation for a further investigation of SOM applications in other variants of routing problems, not only in the polygonal domain but also in high-dimensional spaces with obstacles, where approximate paths between nodes and goals (cities) are necessary, e.g., route planning in 3D environments or in high-dimensional configuration spaces.

Acknowledgement

The work has been supported by the Ministry of Education of the Czech Republic under the program "National research program II" by Project No. 2C06005 and partially by Project No. 7E08006 and EU project No. 216240.

References

[1] B. Angéniol, G. de la C. Vaubois, and J-Y L. Texier. Self-organizing feature maps and the travelling salesman problem. *Neural Networks*, 1: 289–293, 1988.

[2] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. CONCORDE TSP Solver. `http://www.tsp.gatech.edu/concorde.html`, 2003. [cited 8 Jul 2010].

[3] N. Aras, B. J. Oommen, and I. K. Altinel. The Kohonen network incorporating explicit statistics and its application to the travelling salesman problem. *Neural Networks*, 12(9):1273–1284, 1999.

[4] N. Aras, I.K. Altinel, and J. Oommen. A Kohonen-like decomposition method for the Euclidean traveling salesman problem-KNIES_DECOMPOSE. *Neural Networks, IEEE Transactions on*, 14(4):869–890, July 2003.

[5] Yanping Bai, Wendong Zhang, and Zhen Jin. An new self-organizing maps strategy for solving the traveling salesman problem. *Chaos, Solitons & Fractals*, 28(4):1082 – 1089, 2006.

[6] Marco Budinich. A self-organizing neural network for the traveling salesman problem that is competitive with simulated annealing. *Neural Comput.*, 8(2):416–424, 1996.

[7] Laura Burke. "Conscientious" neural nets for tour construction in the traveling salesman problem: the vigilant net. *Computers and Operations Research*, 23(2):121–129, 1996.

[8] Laura I. Burke. Neural methods for the traveling salesman problem: insights from operations research. *Neural Networks*, 7(4):681–690, 1994.

[9] Laura I. Burke and Poulomi Damany. The guilty net for the traveling salesman problem. *Computers and Operations Research*, 19(3-4): 255–265, 1992.

[10] Vašek Chvátal, William Cook, George B. Dantzig, Delbert R. Fulkerson, and Selmer M. Johnson. *Solution of a Large-Scale Traveling-Salesman Problem*, chapter 1, pages 7–28. Springer Berlin Heidelberg, 2010.

[11] E. M. Cochrane and J. E. Beasley. The co-adaptive neural network approach to the Euclidean travelling salesman problem. *Neural Networks*, 16(10):1499–1525, 2003.

[12] Jean-Charles Créput and Abderrafiaa Koukam. A memetic neural network for the Euclidean traveling salesman problem. *Neurocomput.*, 72 (4-6):1250–1264, 2009.

[13] Olivier Devillers, Sylvain Pion, Monique Teillaud, and Projets Prisme. Walking in a Triangulation. *Internat. J. Found. Comput. Sci*, 13: 106–114, 2001.

[14] Masato Edahiro, Iwao Kokubo, and Takao Asano. A new point-location algorithm and its practical efficiency: comparison with existing algorithms. *ACM Trans. Graph.*, 3(2):86–109, 1984.

[15] Jan Faigl. *Multi-Goal Path Planning for Cooperative Sensing*. PhD thesis, Czech Technical University in Prague, 2010.

[16] Jan Faigl. Approximate Solution of the Multiple Watchman Routes Problem with Restricted Visibility Range. *IEEE Transactions on Neural Networks*, 21(10):1668–1679, 2010.

[17] Jan Faigl, Miroslav Kulich, and Libor Přeučil. A sensor placement algorithm for a mobile robot inspection planning. *Journal of Intelligent & Robotic Systems*, 62(3):329–353, 2011.

[18] Jan Faigl, Miroslav Kulich, Vojtěch Vonásek, and Libor Přeučil. An Application of Self-Organizing Map in the non-Euclidean Traveling Salesman Problem. *Neurocomputing*, 74(5):671–679, 2011.

[19] J. C. Fort. Solving a combinatorial problem via self-organizing process: An application of the Kohonen algorithm to the traveling salesman problem. *Biological Cybernetics*, 59(1):33–40, 1988.

[20] Bernd Fritzke and Peter Wilke. FLEXMAP - A Neural Network For The Traveling Salesman Problem With Linear Time And Space Complexity. In *Proc. of IJCNN, Singapore*, pages 929–934, 1991.

[21] H.H. González-Baños, D. Hsu, and J.-C. Latombe. Motion planning: Recent developments. In S.S. Ge and F.L. Lewis, editors, *Autonomous Mobile Robots: Sensing, Control, Decision-Making and Applications*, chapter 10. CRC Press, 2006.

[22] Hui-Dong Jin, Kwong-Sak Leung, Man-Leung Wong, and Z.-B. Xu. An efficient self-organizing map designed by genetic algorithms for the traveling salesman problem. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 33(6):877–888, Dec. 2003.

[23] Marcelo Kallmann. Path Planning in Triangulations. In *Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, Edinburgh, Scotland, July 31 2005.

[24] Miroslav Kulich, Jan Kout, Libor Přeučil, Jiří Pavlíček, and Roman Mázl et al. PeLoTe - a Heterogeneous Telematic System for Cooperative Search and Rescue Missions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems 2004*, volume 1, Sendai, 2004.

[25] Miroslav Kulich, Jan Faigl, and Libor Přeučil. Cooperative planning for heterogeneous teams in rescue operations. In *IEEE International Workshop on Safety, Security and Rescue Robotics*, pages 230–235, 2005.

[26] Kwong-Sak Leung, Hui-Dong Jin, and Zong-Ben Xu. An expanding self-organizing neural network for the traveling salesman problem. *Neurocomputing*, 62:267–292, 2004.

[27] Thiago A. S. Masutti and Leandro N. de Castro. A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem. *Inf. Sci.*, 179(10):1454–1468, 2009.

[28] Kazushi Murakoshi and Yuichi Sato. Reducing topological defects in self-organizing maps using multiple scale neighborhood functions. *Biosystems*, 90(1):101–104, 2007.

[29] M. H. Overmars and E. Welzl. New methods for computing visibility graphs. In *SCG '88: Proceedings of the fourth annual symposium on Computational geometry*, pages 164–171, New York, NY, USA, 1988. ACM.

[30] Alessio Plebe and Angelo Marcello Anile. A neural-network-based approach to the double traveling salesman problem. *Neural Comput.*, 14 (2):437–471, 2002.

[31] Gerhard Reinelt. TSPLIB– A Traveling Salesman Problem Library. *Journal on Computing,*, 3(4):376–384, 1991.

[32] Mitul Saha, Tim Roughgarden, Jean-Claude Latombe, and Gildardo Sánchez-Ante. Planning Tours of Robotic Arms among Partitioned Goals. *Int. J. Rob. Res.*, 25(3):207–223, 2006.

[33] Raimund Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom. Theory Appl.*, 1(1):51–64, 1991.

[34] K. Smith, M. Palaniswami, and M. Krishnamoorthy. Neural techniques for combinatorial optimization with applications. *Neural Networks, IEEE Transactions on*, 9(6):1301–1318, Nov 1998.

[35] Kate A. Smith. Neural Networks for Combinatorial Optimization: a Review of More Than a Decade of Research. *INFORMS J. on Computing*, 11(1):15–34, 1999.

[36] Samerkae Somhom, Abdolhamid Modares, and Takao Enkawa. A self-organising model for the travelling salesman problem. *Journal of the Operational Research Society*, pages 919–928, 1997.

[37] Frederico Carvalho Vieira, Adri ao Duarte Dória Neto, and José Alfredo Ferreira Costa. An Efficient Approach to the Travelling Salesman Problem Using Self-Organizing Maps. *International Journal of Neural Systems*, 13(2):59–66, 2003.

[38] Haiqing Yang and Haihong Yang. An Self-organizing Neural Network with Convex-hull Expanding Property for TSP. In *Neural Networks and Brain, 2005. ICNN&B '05. International Conference on*, volume 1, pages 379–383, Oct. 2005.

[39] Wendong Zhang, Yanping Bai, and Hong Ping Hu. The incorporation of an efficient initialization method and parameter adaptation using self-organizing maps to solve the TSP. *Applied Mathematics and Computation*, 172(1):603–623, 2006.