An Application of the Self-Organizing Map in the non-Euclidean Traveling Salesman Problem

Jan Faigl, Miroslav Kulich, Vojta Vonásek, Libor Přeučil

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

Abstract

An application of the self-organizing map (SOM) to the Traveling Salesman Problem (TSP) has been reported by many researchers, however these approaches are mainly focused on the Euclidean TSP variant. We consider the TSP as a problem formulation for the multi-goal path planning problem in which paths among obstacles have to be found. We apply a simple approximation of the shortest path that seems to be suitable for the SOM adaptation procedure. The approximation is based on a geometrical interpretation of SOM, where weights of neurons represent nodes that are placed in the polygonal domain. The approximation is verified in a set of real problems and experimental results show feasibility of the proposed approach for the SOM based solution of the non-Euclidean TSP.

Keywords: Neural network, Self-organizing map, Traveling salesman problem, Multi-goal path planning

1. Introduction

Self-organizing maps (SOM), also known as the Kohonen-type network, have been applied to solve the Traveling Salesman Problem (TSP) by several approaches during last decades. One of the earliest approaches were proposed in 1988 [1, 2]. After that, enhanced variants of adaptation procedures have been investigated [3, 4]. A notion of inhibition has been proposed to avoid nodes to win too often [5]. In [6], an initialization of neuron weights is discussed and superior results are reported for starting positions of nodes as the convex hull approximation of the cities. Aras et at. used geometrical properties of the ring and a topology of cities in the KNIES algorithm [7]. The convex hull property has been studied in the expanding SOM variant called ESOM [8]. Probably the most complex SOM algorithm for the TSP is the Co-Adaptive net introduced in [9]. A recent variant of the SOM approach with evolutionary principles has been presented in [10]. Even though these approaches increase quality of solutions and are able to solve problems with thousands of cities, they are strictly focused on the Euclidean TSP [11].

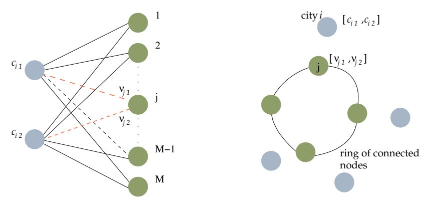
A type of the non-Euclidean TSP is the multi-goal path planning for a mobile robot. The problem is to find a collision-free path that visits a set of locations in the robot workspace [12]. If a point robot can be assumed and the robot workspace can be represented by the polygonal domain $\mathcal{W}$, the shortest-path roadmap approach can be used to find a path between two locations in $\mathcal{W}$ [13, 14]. The difficulty of the SOM application to this kind of TSP is in an efficient determination of node–city distances used in the competitive rule, i.e. a path in order to adapt neurons weights in $\mathcal{W}$. Although the Euclidean distance can be simply and efficiently computed, poor quality solutions are found if only the Euclidean distance is used in the non-Euclidean TSP. Such quality degradation is not SOM specific and it has also been shown for other common construction heuristics [15]. Even for humans the difficulty of solving the TSP with obstacles has been observed, while near optimal solutions have been found for the Euclidean TSP [16]. Difficulty of the shortest path determination is probably the main reason why SOM researchers consider only the Euclidean variant of the TSP.

In this paper, we show an approach based on approximation of the shortest path between a point and city in $\mathcal{W}$ that seems to be sufficient in a SOM based algorithm for the non-Euclidean TSP. Even though the approach is still more computationally demanding than computation of the Euclidean distance, it enables applicability of the SOM in the polygonal domain, because it is significantly faster than naïve approach based on computation of the exact shortest path. Moreover, a valuable benefit of the used approximation is its simplicity, it does not require sophisticated underlying structures like SPM [17]. It is based on a convex polygon partition, but eventually any polygon division can be used. Our prior work in this field has been based on a set of SPM (one for each city) [18], which requires more memory resources than a single polygon division. Besides, a single overlay of the SPMs contains much more cells and it suffers by numerical stability issues.

This paper is organized as follows. A brief description of the selected SOM adaptation procedure [5] is presented in the next section. It contains a geometrical interpretation of the adaptation procedure, which is crucial for the applied path approximation, because it provides a computational geometry point of view to the network evolution in the polygonal domain. The approximate procedure is described in Section 3. Experimental results together with a discussion of future improvements are presented in Section 4. Concluding overview of the presented approach is presented in Section 5.

2. SOM adaptation procedure for the TSP

A structure of the self-organizing neural network for the TSP can be represented as a two-layered competitive learning network [5]. It contains two dimensional input vectors and an array of output units. An association between the learning network and a geometrical representation of the TSP solution is shown in Fig. 1. An input vector i



(a) (b)

Figure 1: Schema of the two-layered neural network and associated geometric representation.

represents coordinates (c_{i1}, c_{i2}) of the city c_i and weights ν_{j1}, ν_{j2} can be interpreted as coordinates of the node ν_j. The network is initialized with small random connection weights and cities are then sequentially applied to the network in a random order. The output nodes compete to be the winner for a given city. The weight vectors of the winner node and its neighbouring nodes are updated in order to get closer to the city according to the neighbouring function f. The nodes form a ring and after the adaptation, the ring represents tour over the applied cities. The network evolves until each city has sufficiently close the winner node (less than given error δ). A schema of the adaptation process is depicted in Algorithm 1.

Algorithm 1: SOM Procedure for the TSP

```
Input: $\boldsymbol{C} = \{c_1, \dots, c_n\}$ - set of cities
Input: $(d, G, \mu, \alpha)$ - parameters of SOM
Input: $\delta$ – maximal allowable error
Input: $i_{max}$ – max. no. of adaptation steps
Output: $(\nu_1, \dots, \nu_m)$ - sequence of node
        weights representing the city tour
init$(\nu_1, \dots, \nu_m)$      // set of neurons weights
$i \leftarrow 0$          // the adaptation step counter
repeat
    $error \leftarrow 0$
    $\boldsymbol{I} \leftarrow \emptyset$          // a set of inhibited nodes
    $\Pi(\boldsymbol{C}) \leftarrow$ a random permutation of cities
    foreach $c \in \Pi(\boldsymbol{C})$ do
        $\nu^\star \leftarrow$ select winner node to $c$, $\nu^\star \notin I$
        $error \leftarrow \max\{error, |\nu^\star, c|\}$
        adapt$(\nu^\star, c)$
        $I \leftarrow I \cup \{\nu^\star\}$    // inhibit winner node
    $G \leftarrow (1-\alpha)G$         // decrease the gain
    $i \leftarrow i + 1$    // increment the step counter
until $error \leq \delta$ or $i \geq i_{max}$
```

An inhibition mechanism is used to associate distinct winner node to each city during one complete presentation of all cities to the network (one adaptation step). A winner node is marked as inhibited and it does not compete to be winner for another city for the rest of the adaptation step. At the end of each adaptation step, a tour can be constructed from the winner nodes by traversing the ring and the exact length of the tour can be found as the sum of city–city lengths.

The winner node is selected according to $\nu^\star = \operatorname{argmin}_\nu |c, \nu|$, where $|.,.|$ denotes the Euclidean distance between the city c and the node ν for the Euclidean TSP. The adaptation function (`adapt`) moves the winner node and its neighbouring nodes towards the presenting city c according to the rule $\nu'_j = \nu_j + \mu f(.,.)(c - \nu_j)$, where μ is the fractional learning rate. Authors of [5] suggested the neighbouring function $f(G, d) = exp(-d^2/G^2)$ for $d < 0.2m$ and $f(G, d) = 0$ otherwise, where G is the gain parameter, d is the cardinal distance measured along the ring and m is the number of nodes. An appropriate initial value of G depends on the size of problem and authors proposed $G_0 = 0.06 + 12.41n$. Recommend values of learning and decreasing rates are $\mu = 0.6$, resp. $\alpha = 0.1$.

An application of the presented adaptation schema to the multi-goal path planning requires nodes in the free space, therefore paths (and distances) have to respect obstacles. The efficiency of the SOM algorithm relies on determination of the winner node, which uses a node–city distance. The winner node and its neighbouring nodes are then moved closer to the city along the path avoiding obstacles, i.e. a node weights are set to a new position at the path such that the node travels a distance $\mu f(G, d)|S|$ traversing the path, where $|S|$ is the node–city distance. Because a winner node is selected to each city from the set of all nodes in each adaptation step, the efficient determination of the collision-free path is crucial for a reasonable computational requirements of the SOM adaptation procedure.

3. Approximation of the Shortest Path in the SOM based TSP

During the adaptation process, weights (nodes) are modified in order to find a solution of the TSP. The modification process can be viewed as an exploration of the problem domain. In the first adaptation steps, nodes are moved across the problem domain, while in the final steps, due to lower value of the gain G, weights (nodes) changes are very small. Such observations can lead to expect that a rough approximation of the shortest path in early steps should be sufficient, because position of the nodes is dramatically changed during movements. On the other side, the winner nodes are very close to the cities in final steps. An example of the performance of the SOM adaptation in an environment with obstacles is shown in Fig. 2[1]. These observa-

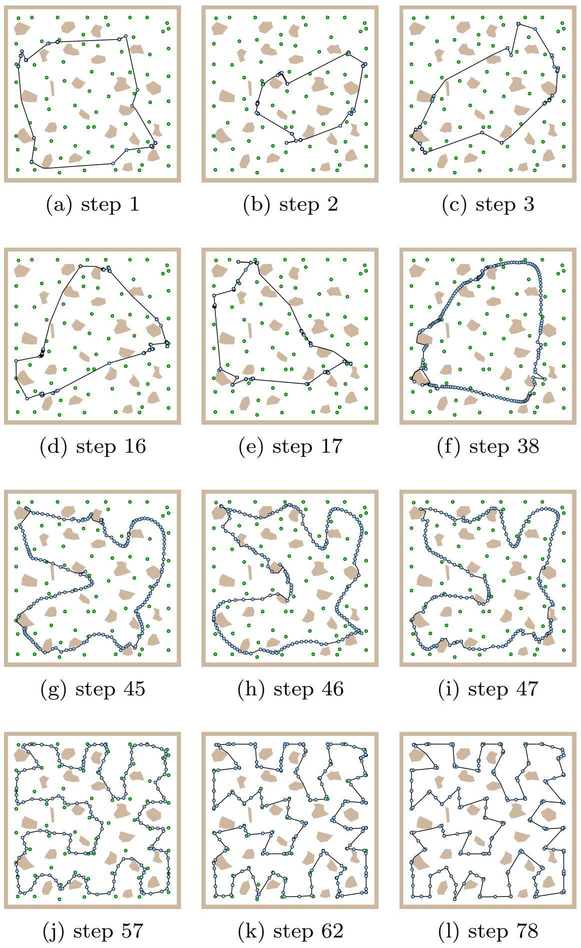

(a) step 1 (b) step 2 (c) step 3
(d) step 16 (e) step 17 (f) step 38
(g) step 45 (h) step 46 (i) step 47
(j) step 57 (k) step 62 (l) step 78

Figure 2: An example of performance of the adaptation procedure in the TSP with obstacles, small standalone disks represent cities and connected disks represent nodes (weights) that form a ring.

tions are main motivation for the following application of approximate shortest path in the polygonal domain $\mathcal{W}$.

The approximation is based on a convex polygon partition of $\mathcal{W}$, which can be found in $O(v \log v)$, where v is the number of vertices of $\mathcal{W}$ [19]. The convex polygon partitioning divides free space of $\mathcal{W}$ into a set of convex cells. A node is always placed in the free space, thus it is always placed in some cell. The shortest path from a vertex of the cell to the city can be used as approximation of the path from a node to the city. The shortest paths from vertices to cities can be found in the visibility graph by Dijkstra's algorithm in time $O(ne \log(v + n))$, where n is the number of cities, v is the number of

[1]The nodes are connected by black line segments just to show neighbours of the nodes. It is not necessary to compute a path between two nodes in the TSP.

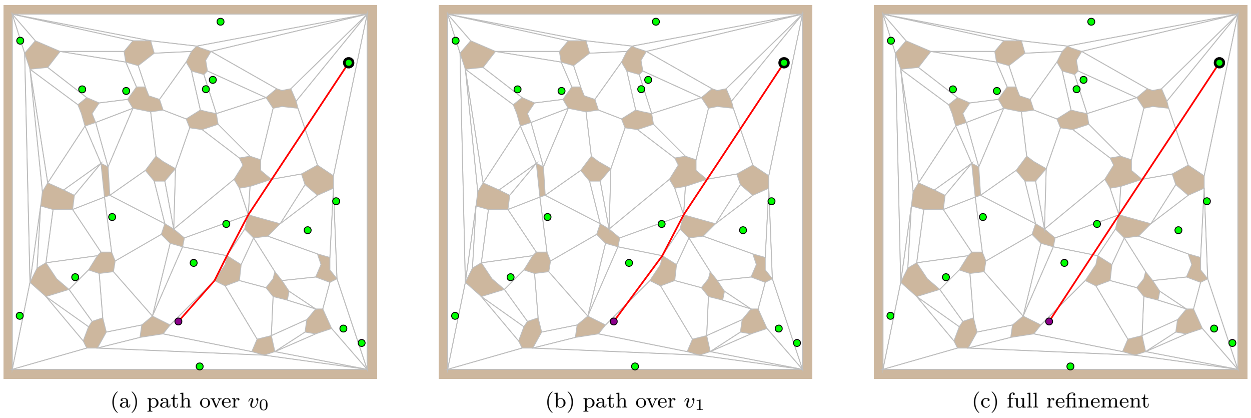

(a) path over v_0 (b) path over v_1 (c) full refinement

Figure 3: An example of path refinement, gray segment represents diagonals of the convex partition, small disks are cities, a node is connected with the city by the approximation of the shortest path.

vertices and e is the number of visible pairs (city–city, city–vertex, and vertex–vertex), which can be bounded $e \leq v + vn$. The full visibility graph can be found in $O((v+n)^2)$ [20].

More formally, assume a polygonal workspace $\mathcal{W}$ with v vertices and let $\mathbf{P}$ be a convex polygon partition of $\mathcal{W}$ into convex cells C_i, $\mathbf{P} = \{C_1, C_2, \ldots, C_k\}$, where each cell C is represented as a sequence of polygon vertices, and assume a node ν in the cell C_ν. The approximate path from ν to the city c is found as the shortest path $S(w, c)$ over vertex w of C_ν to c such that $w = \mathrm{argmin}_{w_i \in C_\nu} |\nu, w_i| + |S(w_i, c)|$, where $|.,.|$ denotes the Euclidean distance between two points and $|S(.,.)|$ is the length of the shortest vertex–city path.

The problem to find the cell C_ν is the point–location problem, which can be solved in $O(\log v)$ [21]. The complexity of the path determination depends only on the number of cells vertices and the adaptation of node depends only on the number of vertices, because paths from each vertex to all cities can be pre-computed and a node is then moved along the path with v vertices at maximum.

A precision of the proposed approximation depends on the size of the convex cells and can be insufficient for an arbitrary point. An initial path can be improved by the following refinement procedure. Assume a node ν inside the cell C_ν and approximation of the path from ν to the vertex v_k as a sequence of vertices $(v_0, v_1, \ldots, v_k)$, $v_0 \in C_\nu$. A refinement is an examination of direct visibility test between ν and v_i:

```
1. $i \leftarrow 0$
2. **while** visible$(\nu, v_{i+1}) \wedge i < k$ **do**
     $i \leftarrow i + 1$
3. $path \leftarrow (v_i, v_{i+1}, \ldots v_k)$.
```

The visibility test can be based on the straight walk procedure [22]. If a straight line from ν to the vertex v_k intersects only diagonals or entirely lies 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. 3.

The path refinement can be performed up to the selected number of vertices. The higher number increases the computational burden, but a shorter path can be found. The effect of the path refinement is experimentally verified for a set of selected maximal number of examined vertices in Section 4. The full path refinement procedure is denoted as *pa* algorithm variant and *va-j* denotes refinement up to j vertices. It means that the *va-0* algorithm variant uses the initial path, and only one vertex (the second one[2]) is examined in the *va-1* variant.

4. Experimental Results

Performance of the used SOM based algorithm for the TSP has been evaluated in a set of TSPLIB problems [23] prior the evaluation of the approximation of the shortest path. Mainly to find an estimation of the algorithm quality without effect

[2]Vertex v_0 is part of the C_ν cell, so it is directly visible from ν.

of the path approximation. SOM is a randomized algorithm therefore each problem has been solved 20 times by the particular algorithm variant. The recommended parameters of the SOM procedure [5] are used as they have been presented in Section 2. The number of neurons has been set to $2.5n$, where n is the number of cities. The adaptation has been terminated if the *error* is less than 0.001. The quality of solutions is evaluated as the percent deviation to the optimum tour length of the mean solution value, $PDM = (\overline{L} - L_{opt})/L_{opt} \cdot 100\%$, and as the percent 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 [24]. The selected TSPLIB problems represent instances of the Euclidean TSP therefore the Euclidean distance is used in the adaptation procedure. Quality metrics provide estimation of algorithm performance and can be used to evaluate influence of the path approximation. Results for selected problems are presented in Table 1. The average value of the PDM is about 5%.

Table 1: Performance of the SOM algorithm on TSPLIB problems

Problem	N_C	L_{opt} [m]	**PDM**	**PDB**	T [s]
eil51	51	4.3	3.8	2.5	0.02
berlin52	52	75.4	6.9	0.0	0.03
st70	70	6.8	2.6	1.2	0.04
eil76	76	5.4	5.0	2.9	0.05
rd100	100	79.1	4.0	1.2	0.10
eil101	101	6.4	5.2	2.9	0.09
bier127	127	1182.9	4.1	1.7	0.17
ch130	130	61.1	3.8	2.3	0.17
rat195	195	23.3	8.7	6.6	0.37
kroB200	200	294.4	4.1	1.5	0.42
gil262	262	23.9	5.5	4.4	0.70
lin318	318	420.4	4.8	3.4	1.17
rd400	400	152.8	5.4	3.9	1.86
u574	574	369.3	5.7	4.8	4.07
rat575	575	68.0	7.2	6.3	3.91
rat783	783	88.5	7.3	6.1	7.46

The proposed approximation of the node–city path has been experimentally verified in a set of problems with obstacles. Due to lack of common problems for environments with obstacles a set of environments (in a form of polygonal maps) used in the motion planning has been utilized. Parameters of the environments are shown in Table 2, where N_V is the number of vertices, N_H is the number of holes and N_R is the number of convex cells (of the supporting convex partition), environments *a*, *jh*, *pb*, *ta*, and *h2* represent maps of real buildings. The cities are found as sets of sensing locations in

Table 2: Testing environments with obstacles

Name	**Dimensions** [m × m]	N_V	N_H	N_R
jari	4.5 × 4.9	48	1	14
complex2	20.0 × 20.0	40	3	21
m1	4.8 × 4.8	51	4	26
m2	4.8 × 4.8	51	6	20
map	4.8 × 4.8	68	8	36
potholes	20.0 × 20.0	153	23	75
rooms	20.0 × 20.0	80	0	33
a	8.9 × 14.1	99	6	22
dense	21.0 × 21.5	288	32	150
jh	20.6 × 23.2	196	9	77
pb	133.3 × 104.8	89	3	41
ta	39.6 × 46.8	74	2	30
h2	84.9 × 49.7	2062	34	476

the inspection task, like in [13] and together with maps of real environments they provide realistic size of the problems. The name of the environment with a subscript denoting the visibility range in meters for the inspection is used as the problem name of the particular TSP. Cities of problems without the subscript have been found as sensing locations for the unrestricted visibility range.

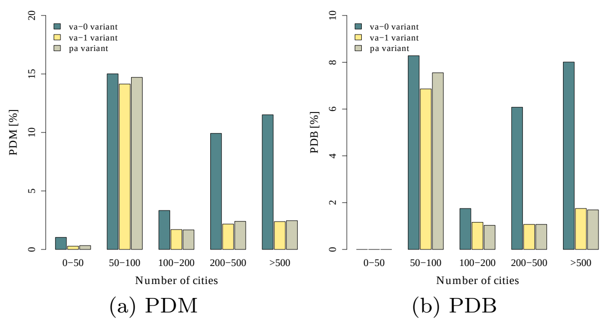
Because of the proposed approximation of the shortest path, a found path can be different from the exact path, which can cause that a stable solution with the required *error* is not found. Therefore the adaptation procedure is terminated after 180 adaptation steps.

The path approximation is evaluated for the three variants of the path refinement: without the refinement (*va-0*), examination of the second vertex of the path (*va-1*) and the full path refinement (*pa*). Detail results are presented in Table 3, where N_C

Table 3: Comparison of refinement variants

Problem	N_C	L_{opt} [m]	***va-0* variant** PDM	PDB	T [s]	***va-1* variant** PDM	PDB	T [s]	***pa* variant** PDM	PDB	T [s]
jari	6	13.6	0.84	0.00	0.07	0.21	0.00	0.08	0.10	0.00	0.08
complex2	8	58.5	2.36	0.00	0.09	0.29	0.00	0.10	0.00	0.00	0.11
m1	13	17.1	4.77	0.00	0.12	0.14	0.00	0.16	0.19	0.00	0.18
m2	14	19.4	11.17	6.33	0.12	12.51	8.95	0.17	11.23	7.33	0.19
map	17	26.5	4.87	0.00	0.18	3.85	0.00	0.25	3.24	0.00	0.28
potholes	17	88.5	1.61	0.00	0.32	1.48	0.00	0.33	1.09	0.00	0.40
a	22	52.7	1.03	0.00	0.28	0.28	0.00	0.40	0.33	0.00	0.56
rooms	22	165.9	1.55	0.11	0.29	1.01	0.17	0.44	0.85	0.00	0.46
dense$_4$	53	179.1	15.01	8.28	2.54	14.14	6.86	2.71	14.71	7.55	3.28
potholes$_2$	68	154.5	13.77	9.67	3.02	5.06	3.09	3.54	5.25	2.97	4.81
jh$_2$	80	201.9	7.45	4.73	2.39	2.04	0.43	4.96	1.74	0.87	6.94
pb$_4$	104	654.6	4.35	1.82	3.37	1.10	0.03	6.88	0.60	0.04	7.89
ta$_2$	141	328.0	8.44	5.90	5.81	2.96	2.13	11.94	3.29	2.05	14.49
h2$_5$	168	943.0	3.33	1.75	33.79	1.70	1.16	67.68	1.67	1.03	92.78
potholes$_1$	282	277.3	30.73	25.95	32.52	6.54	3.75	64.60	6.05	4.03	91.63
pb$_{1.5}$	415	839.6	9.92	6.08	57.08	2.17	1.07	123.92	2.40	1.07	140.04
h2$_2$	568	1316.2	11.51	8.01	321.27	2.38	1.75	686.24	2.46	1.69	958.37
ta$_1$	574	541.1	13.41	12.03	106.75	5.46	4.77	225.98	5.35	4.59	275.90

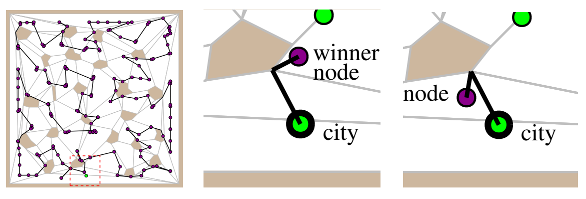
is the number of cities. Notice the computational time T for problems $h2_2$ and ta_1, the *h2* environment contains more vertices, therefore it is more computationally demanding. An overview of the quality metrics is shown as histograms in Fig. 4. Selected found solutions are presented in Fig. 5.



(a) PDM (b) PDB

Figure 4: Quality of solution.

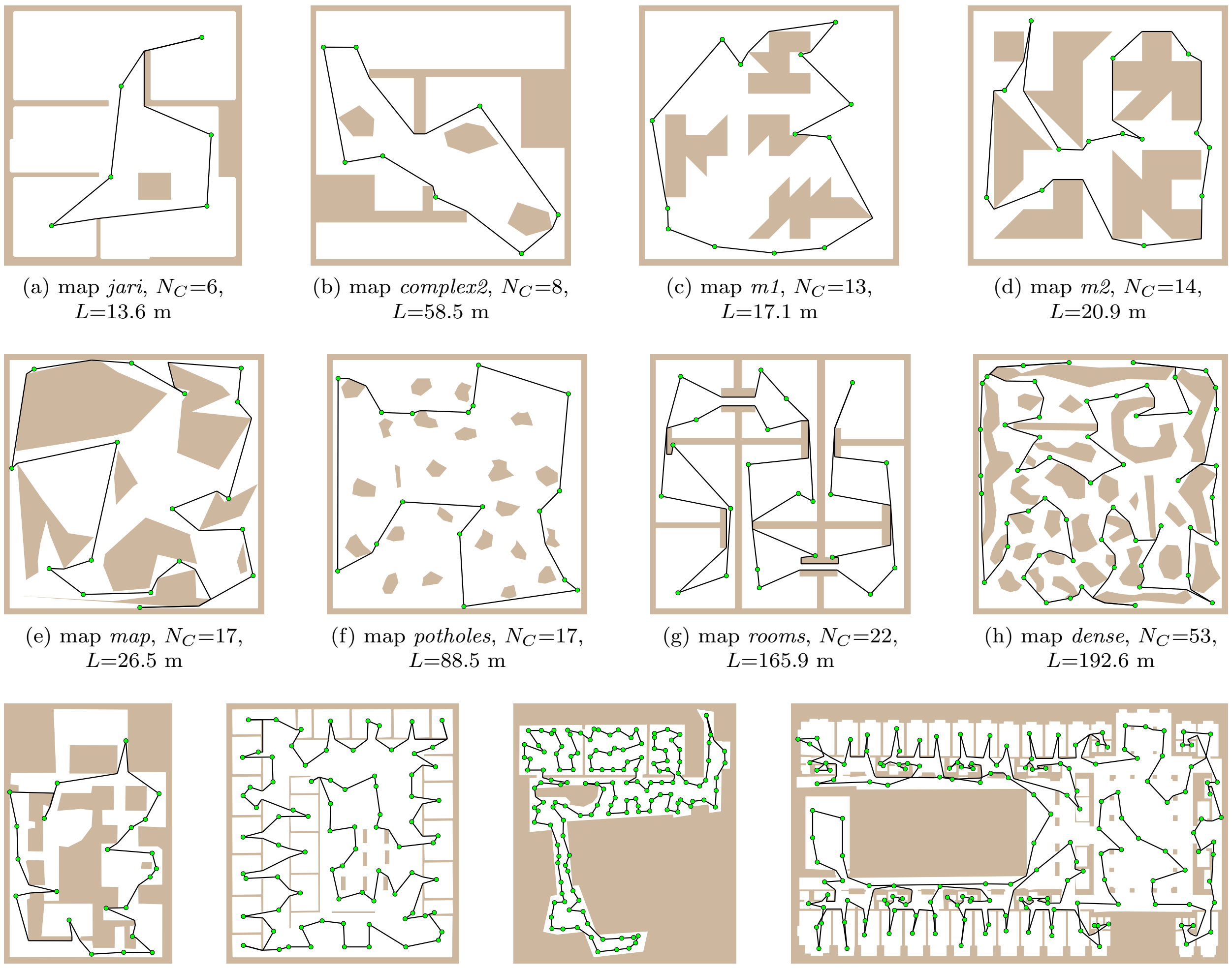
An error of the path approximation is too high for the *va-0* variant and the adaptation has been terminated after 180 steps in several cases. Such termination has been observed for eight environments, mostly for the *potholes*, *dense*, and *m2* environments. The convergence issue has not been observed if a refinement of the path has been performed. The cause of the issue is demonstrated in Fig. 6. One city is not covered by the winner



(a) convergence issue (b) node entering to the next (city) cell (c) node in the next (city) cell

Figure 6: Effect of error of the *va-0* path approximation, map *potholes*.

node, because the winner is not effectively moved towards the city. For the *va-0* variant, the path refinement is not performed and the length of the path over the cell vertex is of course longer than the



(a) map *jari*, N_C=6, L=13.6 m
(b) map *complex2*, N_C=8, L=58.5 m
(c) map *m1*, N_C=13, L=17.1 m
(d) map *m2*, N_C=14, L=20.9 m
(e) map *map*, N_C=17, L=26.5 m
(f) map *potholes*, N_C=17, L=88.5 m
(g) map *rooms*, N_C=22, L=165.9 m
(h) map *dense*, N_C=53, L=192.6 m
(i) map *a*, N_C=22, L=52.7 m
(j) map *jh*, N_C=80, L=203.7 m
(k) map *ta*, N_C=141, L=334.8 m
(l) map *h2*, N_C=168, L=952.6 m

Figure 5: Selected best found solution by the *pa* algorithm variant.

Euclidean distance (in this case the node and the city are directly visible). All nodes except this one are sufficiently close to cities (in distance less than *error*), but the node is not effectively moved due to small value of the gain G. The higher value of the gain does not really help, because a winner node of the city can be moved to the next city cell during its movement along the shortest path over a cell vertex, see Fig. 6b. Once the node is in the next cell, the length of the approximate path is longer than before and the current value of G is insufficient. This is a cost of the independent construction of the polygon partition without relation to the cities' positions. In this particular case, the convergence issue does not affect the solution quality, because the winner node is associated with the "right" city. For cases with more cities the issue can decrease the quality of found solutions and the path refinement is useful.

The results show that the *va-1* variant is sufficient and full path refinement does not significantly improved the quality of found solutions. The overall quality of solutions is competitive with the Euclidean distance in the TSPLIB problems.

4.1. Required computationl time

All results presented in this paper have been performed within the same computational environment: the MAC Mini with the Intel P7350@2 GHz CPU, 2 GB RAM running FreeBSD 7.2, and only one CPU core has been utilized during experiments. Therefore required computational times of the algorithm variants can be directly compared. The algorithms have been implemented in C++ and compiled by the G++ 4.2 with -O2 optimization flag. The computational time depends on the number of cities and also on the particular environment (the number of vertices and the number of convex cells). A comparison of the required computational times for particular refinement variants is shown in Fig. 7 as histograms of the average values of the computational times.

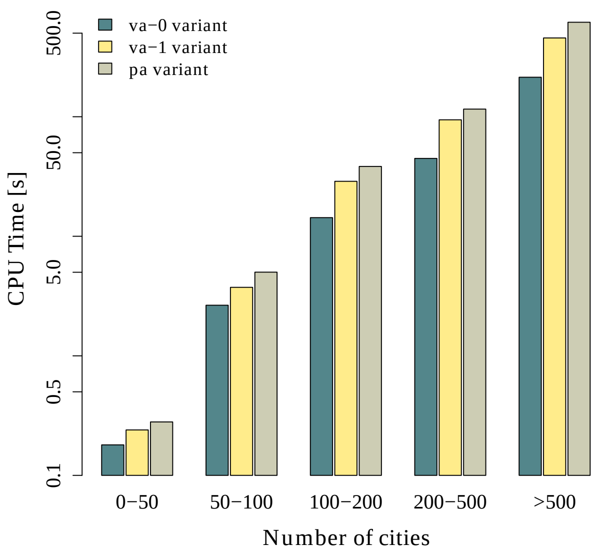


Figure 7: Required computational time.

In order to use the presented approximation, two supporting structures have to be pre-computed. The required computational time to create a convex polygon partition is in units or tens of milliseconds and it is negligible in comparison to the required time of the adaptation procedure. Also construction of the visibility graph is very fast, it is mostly found in tens of milliseconds and in 154 milliseconds for the largest problem with 586 cities and 2062 vertices ($h2_2$). The most time expensive part of the preparation phase is a computation of the shortest path between cities (and vertices). Even though the time represents fraction of the computational requirements of the adaptation procedure, it is included in the presented results.

The used point location algorithm is based on the interval trees [21] and it provides thousands of queries per millisecond in the used computational environment. Particularly 4950 queries per millisecond (qpms) for the environment *jh*, 2145 qpms for *m3*, and 1736 qpms for *h2*.

4.2. Comparison with Other Approaches

A comparison of the proposed algorithm with other approaches is not an easy task, because of the two following reasons. The first one is lack of SOM approaches for the TSP in the polygonal domain. In our best knowledge, only the SOM algorithm for a graph input [25] deals with obstacles in the TSP and can be used for the multi-goal path planning. Even though the authors demonstrated the approach in a TSP with obstacles, the paper lacks serious performance evaluation. Therefore a set of experiments have been performed for various densities of a triangular mesh that forms a graph used in their algorithm and problems with different numbers of cities. The proposed algorithm based on a convex partition provides better results than the graph based algorithm for problems with small number of cities (less than hundreds). The computational requirements are competitive. For problems with higher number of cities the proposed algorithm is significantly faster than the graph based algorithm, because the graph (triangular mesh) have to be dense in order to find a solution with competitive quality.

The second difficulty of the comparison is relatively poor performance of SOM regarding the performance of the heuristics from the Operational Research (OR). It is a known fact that the effective implementation of the Lin-Kernighan heuristic [26] provides superior quality in a lot of problems. It is also the case for the used problems in this paper and the *linkern* [27] solver from the Concorde package [24]. The *linkern* solver provides solutions typically in hundreds of milliseconds[3] and with the PDM quality in tenths of percent. The largest problem with obstacles $h2_2$ is solved in four seconds with the average quality around two-tenths. These results are grounds for pessimism, which also the case for the Euclidean TSP. Authors of [9] noted that to make the SOM competitive to OR approaches the computational time should be improved by three orders of magnitude and the solution quality by one order of magnitude. For the non-Euclidean TSP the

[3]Here, solutions are found in a pre-computed graph of all shortest path between cities.

Table 4: Comparison with the GENI algorithm

Problem	SOM		GENI	
	PDM	T [s]	**PDM**	T [s]
jari	0.10	0.08	0.00	0.01
complex2	0.00	0.11	0.00	0.01
m1	0.19	0.18	0.00	0.03
m2	11.23	0.19	1.36	0.03
map	3.24	0.28	0.81	0.07
potholes	1.09	0.40	0.34	0.13
a	0.33	0.56	0.00	0.16
rooms	0.85	0.46	0.00	0.15
dense_4	14.71	3.28	4.63	1.27
$\mathrm{potholes}_2$	5.25	4.81	2.98	1.81
jh_2	1.74	6.94	0.75	2.53
pb_4	0.60	7.89	0.11	4.15
ta_2	3.29	14.49	1.86	8.02
$\mathrm{h2}_5$	1.67	92.78	0.88	14.30
$\mathrm{potholes}_1$	6.05	91.63	4.44	31.49
$\mathrm{pb}_{1.5}$	2.40	140.04	0.82	68.05
$\mathrm{h2}_2$	2.46	958.37	1.32	135.08
ta_1	5.35	275.90	3.66	134.41

computational time should be improved even more. It is clear that such daunting tasks will not be accomplished in the near future without sudden major breakthroughs [9]. Therefore we consider "less powerful" state-of-the art construction heuristic to encourage the SOM community and to show that also the OR approaches evolved during the time. The selected heuristic is the GENI solver [28] and its average performance over 20 runs for each problem is depicted in Table 4. The presented computational times T include the time to determine all shortest path between cities.

4.3. Discussion

The examined problems can be considered as relatively small in comparison to the current direction of the SOM application in the TSP with thousands of cities, e.g. [10]. The presented approach is likely to be computationally unfeasible for such large problems. On the other side, from the practical point of view, problems of the path planning for a robot seem to be smaller. In [29], authors reported that a typical number of cities is in tens and less than one hundred. Also in the inspection planning, the number of cities is in hundreds for real sized environments [30]. So, at least for these types of problems, the research can be focused on smaller problems. The proposed approximation can be motivation for further investigation of the SOM application to these problems, which seems to be omitted by the community, probably due to difficulty of determination of the path among obstacles.

The presented results show that the full path refinement does not significantly improve the quality of found solution. Thus, according to the presented results the approximation of the shortest path seems to be sufficient. It means that further investigation of the non-Euclidean problems that requires computationally demanding determination of the distance between presented vector and weights of the network can be based on approximation, which can be significantly faster. For generalized problem in the 3D, approximation of the shortest path is necessary, as the general problem of finding the shortest path among obstacles in the 3D is known to be NP-hard [31].

Although the used approximation of the shortest path is significantly faster than a naïve exact path determination, it is still hundred times slower than a simple computation of the Euclidean distance. The additional speedup improvements may be possible in two different ways. In the first way, more sophisticated geometrical structures can be used and the winner searching process can be informed by a rough estimation of the distance to avoid computationally intensive path determination. The second way is improvement of the used SOM rules that affected the number of node–city path queries. For an example the winner selection can be restricted to a smaller set like in [9], also the number of winner neighbouring nodes involved in the adaptation may be smaller. Finally, the number of queries proportionally depends on the number of adaptation steps. It has to be noted that such restrictions decrease the required computational time, but they also decrease the quality of found solutions. The key problem of the future research in this field is to find adaptation rules that will decrease the number of queries, while the solution quality is preserved or even improved.

5. Conclusion

Difficulty of SOM application to the non-Euclidean TSP has been noted by several authors of SOM algorithms for the TSP. The presented experimental results show that even a simple approximation of the shortest path among obstacles can be used and the SOM algorithm is able to find a solution with competitive quality to a solution of the Euclidean TSP of a similar problem size. The used approximation of the shortest path is more computationally intensive than computation of the Euclidean distance. However, the main result of the presented approach is the feasibility of the approximation that allows eventual application in other problems where approximation is the only way to find a solution with reasonable computational requirements.

The proposed approximation is based on geometrical interpretation and application of structures and algorithms from the computational geometry (CG) domain that can be considered as relatively far from the neural network domain. The necessary algorithms are as follows:

- a visibility graph algorithm,
- a convex polygon partitioning algorithm,
- a point location algorithm,
- a straight walk procedure in a convex partition.

The advantage of the proposed solution is that these algorithms are relatively easy to implement, or they can be substituted by pre-computed structures[4] (visibility graph and polygon partition).

The proposed solution shows a combination of SOM principles with approaches from the CG domain and it can open future research in SOM application to other problems of the CG domain, e.g. Touring a Sequence of Polygons, View Planning Problem, Vision Points, Watchman Route Problem.

Acknowledgement

The work has been supported by the Ministry of Education of the Czech Republic under program "National research program II" by the Project No. 2C06005, and partially by the Project No. 6840770038.

[4]The presented problems in this paper with all necessary supporting structures are available at `http://purl.org/faigl/tsp`.

References

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

[2] 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) (1988) 33–40.

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

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

[5] S. Somhom, A. Modares, T. Enkawa, A self-organising model for the travelling salesman problem, Journal of the Operational Research Society (1997) 919–928.

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

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

[8] K.-S. Leung, H.-D. Jin, Z.-B. Xu, An expanding self-organizing neural network for the traveling salesman problem, Neurocomputing 62 (2004) 267–292.

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

[10] J.-C. Créput, A. Koukam, A memetic neural network for the Euclidean traveling salesman problem, Neurocomputing 72 (4-6) (2009) 1250–1264.

[11] C. H. Papadimitriou, The Euclidean travelling salesman problem is NP-complete, Theoretical Computer Science 4 (3) (1977) 237 – 244.

[12] L. B. Gueta, R. Chiba, J. Ota, T. Ueyama, T. Arai, Coordinated motion control of a robot arm and a positioning table with arrangement of multiple goals, in: Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA), 2008, pp. 2252–2258.

[13] T. Danner, L. Kavraki, Randomized Planning for Short Inspection Paths, in: Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA), San Fransisco, CA, 2000, pp. 971–976.

[14] S. N. Spitz, A. A. G. Requicha, Multiple-Goals Path Planning for Coordinate Measuring Machines, in: Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA), 2000, pp. 2322–2327.

[15] F. Glover, G. Gutin, A. Yeo, A. Zverovich, Construction heuristics for the asymmetric tsp, European Journal of Operational Research 129 (3) (2001) 555 – 568.

[16] J. Saalweachter, Z. Pizlo, Non-euclidean traveling salesman problem, Optimization and Its Applications 21 (2008) 339–358.

[17] J. Hershberger, S. Suri, An Optimal Algorithm for Euclidean Shortest Paths in the Plane, SIAM J. Comput. 28 (6) (1999) 2215–2256.

[18] M. Kulich, J. Faigl, L. Přeučil, Cooperative planning for heterogeneous teams in rescue operations, in: IEEE International Workshop on Safety, Security and Rescue Robotics, 2005, pp. 230–235.

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

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

[21] F. P. Preparata, M. I. Shamos, Computational Geometry: An Introduction (Monographs in Computer Science), Springer, 1985.

[22] O. Devillers, S. Pion, M. Teillaud, P. Prisme, Walking in a Triangulation, Internat. J. Found. Comput. Sci 12 (3) (2002) 193–205.

[23] G. Reinelt, TSPLIB– A Traveling Salesman Problem Library, Journal on Computing, 3 (4) (1991) 376–384.

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

[25] T. Yamakawa, K. Horio, M. Hoshino, Self-Organizing Map with Input Data Represented as Graph, in: ICONIP (1), 2006, pp. 907–914.

[26] K. Helsgaun, An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic, European Journal of Operational Research 126 (1).

[27] D. Applegate, W. Cook, A. Rohe, Chained Lin-Kernighan for Large Traveling Salesman Problems, INFORMS J. on Computing 15 (1) (2003) 82–92.

[28] M. Gendreau, A. Hertz, G. Laporte, New insertion and postoptimization procedures for the traveling salesman problem, Oper. Res. 40 (6) (1992) 1086–1094.

[29] M. Saha, T. Roughgarden, J.-C. Latombe, G. Sánchez-Ante, Planning Tours of Robotic Arms among Partitioned Goals, Int. J. Rob. Res. 25 (3) (2006) 207–223.

[30] J. Faigl, M. Kulich, L. Přeučil, A sensor placement algorithm for a mobile robot inspection planning, Journal of Intelligent & Robotic Systems doi:10.1007/s10846-010-9449-0.

[31] J. Canny, J. Reif, New lower bound techniques for robot motion planning problems, Annu. IEEE Symp. on Foundations of Computer Science (1987) 49–60.