Multiple traveling salesmen problem with hierarchy of cities in an inspection task with limited visibility

Jan Faigl[1], **Miroslav Kulich**[2], **Libor Přeučil**[1]
[1]The Gerstner Laboratory for Intelligent Decision Making and Control
[2]Center of Applied Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
Technická 2, Prague 6, 166 27
Czech Republic
{xfaigl,kulich,preucil}@labe.felk.cvut.cz

Abstract - *This paper introduces a hierarchical self-organizing neural network solving multiple traveling salesmen problem in an inspection tasks with limited visibility in an environment with obstacles. The working environment is represented as a polygon with holes. The inspection task is decomposed into two sub-problems: a) generation of sensing locations (Art Gallery Problem - AGP) and b) connecting of found locations by a set of paths (Multiple Traveling Salesmen Problem - MTSP). Both sub-problems are NP-hard and therefore algorithms finding approximate solutions are used. The AGP solver is based on randomized sampling of the environment, while a self-organizing neural network solves the MTSP. Moreover, the visibility range is limited in real applications. Low visibility leads to increase of number of sensing locations and therefore increases time costs of the MTSP planning. The proposed approach constructs a hierarchical set of sensing locations that allows stepwise learning process of the neural network. The neural net previously learned for limited number of sensing locations is used to enforce learning for higher number of sensing locations.*

Key words - Inspection Task, Robot Path Planning, Multiple Traveling Salesmen Problem

1 Introduction

An inspection task is one of studied problems upon robot path planning. An application of the inspection task can be in search and rescue scenario, in which possible victims have to be found [1]. The problem is to find a route such that a robot surveys whole working space while it moves along the route. The given problem can be constrained so, that the environment is a priori known and it is represented as a polygon with holes. Moreover, a visibility range of the robot can be limited. Considering more complex problem when multiple robots operate in the same environment two criteria evaluating optimality of the found solutions can be

This research is sponsored by grant CTU 0505413
The support of the Ministry of Education of the Czech Republic, under the Project No. 1M6840770004 to Miroslav Kulich is also gratefully acknowledged.

defined. The *MinSum* criterion is denoted as a total length of the routes, while the *MinMax* criterion requires to minimization of the longest route.
Several existing approaches are based on decomposition of the search problem into two subproblems, determination of sensing locations and planning routes over found points. The sensing locations are places in the environment where sensing (search) is performed to observe all points of the environment. The first problem is also known as *Art Gallery Problem* (AGP) [2] and the second problem is *Multiple Traveling Salesman Problem* (MTSP) [3], [4]. Both problems are known as NP-hard. The Art Gallery Problem for a polygon with holes P is more precisely defined as the problem of finding a minimum set of points G (guards) in P such that every point in P is visible from some point of G. If sensing capability of robots is limited to distance d, we can talk about d-visibility (we say that two points in P are d-visible if they are visible and their distance is less than d). An approximation solution of the AGP can be based on randomized dual sampling schema [2] that typically finds less number of sensing locations than theoretical bound.
However a transformation from MTSP to TSP was proposed in [5], the solution of the transformed problem is highly degenerated for the MinMax criterion of the original MTSP problem [6]. That is why the presented approach solves MTSP directly.
The rest of this paper is organized as follows. An algorithm finding sensing locations is described in section 2. A self-organizing neural network approach for the MTSP for polygon with holes is presented in section 3. Reduction of time requirements of the neural network algorithm for high number of sensing locations is described in section 4 followed by experimental results.

2 Determination of sensing locations

The implemented algorithm is based on modified randomized dual sampling of the environment [7]. The inputs of the algorithm are a visibility radius of robot d and a *map* of the environment represented as a polygon with holes and a number of randomly placed points n. The output of the algorithm is a set of sensing locations $\boldsymbol{C}$ which contains locations necessary to complete survey of the whole environment having the limited visibility distance d. The algorithm incrementally adds points to the map while a volume of non-covered region S is larger than 0. In the initial step, the non-covered region S is the whole free-space of the map. Afterwards in the placement procedure, a random point p_b at the border of a region S is placed randomly and the visibility region of p_b in *map* is computed (denoted as $\Lambda(p_b)$). This visibility region is then cropped by the visibility distance d, $V_{p_b} = \Upsilon^d(\Lambda(p_b))$. A given number of points are placed into the cropped visibility region $p_i = \rho(V_{p_b})$, $i = 1, \ldots, n$. For each placed point a cropped visibility region in *map* is computed $V_i = \Upsilon^d(\Lambda(p_i))$. From these points a sensing location $p^\star$ is selected according to a maximal area of the region V_i. The covered region from point $p^\star$ is subtracted from the not covered region S. The placement procedure is repeated until the whole environment is covered.

3 Route finding

Sensing locations found by the algorithm described above stand for the input cities for the MTSP algorithm. A self-organizing neural network algorithm [3] extended for environments

with obstacles [4] was used. The algorithm finds an approximate solution of the Euclidean instances of the MTSP for m salesmen. A route of each salesmen starts and finishes at the same city called *depot*. The main idea of the algorithm is to represent a path of each particular robot (salesman) by a ring of neurons, where neighboring neurons are connected. A ring of connected neurons is generated around depot for each salesman. A number of neurons in the ring is $M \geq \frac{2 \cdot n}{m}$, where n is number of cities and m is number of salesmen (ring).

The algorithm consists of two main steps. The first is an *adaptation* procedure of the closest neuron from each ring to the depot, while an *adaptation* of the closest neuron to each city is performed in the second step. The shortest path between the neuron $\nu_{i,j}^k$ and the city c is weighted according to difference from an average length of rings : $distance(\nu_{i,j}^k, c) = |\nu_{i,j}^k - c| \cdot \left(1 + \frac{len(\boldsymbol{r}_i) - avg}{avg}\right)$, where $\boldsymbol{r}_i$ is i-th ring, $len(\boldsymbol{r}_i)$ is length of the ring $\boldsymbol{r}_i$ and avg is an average length of the rings. The operation $adaptation(\nu_{i,j^\star}^k, c)$ is a movement of the neurons $\nu_{i,j}^k$ from neighborhood $\nu_{i,j^\star}^k$ such that $\nu_{i,j}^{k+1} = \nu_{i,j}^k + \mu \cdot e^{-\frac{d^2}{G^2}} \cdot |\nu_{i,j}^k - c|$, where $d = min\{|j^\star - j|, M - |j^\star - j|\}$, $j^\star$ is an index of the closest neuron in the i-th ring to the city c. A value of G is decreased in each iteration loop by $G = (1 - \alpha)G$. Initial value of G and μ, α are parameters of neural network. The number of neurons in the neighborhood depends on the number of neurons in the ring M, it was selected $\frac{M}{5}$. The algorithm terminates whenever all the cities have a unique closest neuron and maximum distance between the city and its closest neuron is smaller than the minimal distance between cities.

A crucial point of the algorithm is determination of distances between a neuron and a city. For environments without obstacles this is can be achieved by evaluation of Euclidean distance. In cases when obstacles are present in the environment, the distance between two points have to be computed with respect to the obstacles. Such distance is called *Euclidean geodesic*. A suitable structure for effective determination of the shortest distance between city and neuron is *Shortest Path Map* (SPM). A sub-quadratic algorithm for computing the SPM presented in [8] was used.

The second issue of the algorithm for a polygon with holes is determination of the length of the ring. Therefore, instead of computing lengths of rings directly, the length of the ring is computed as a length of a path traveled by each salesman that visits cities along the ring. The nearest neuron is determined for each city C_i and all neurons in the order they appear in the ring are sequentially examined. If the neuron is nearest to some city this city is added into a sequence of cities visited by the salesman. The complete length of the ring is then defined as a sum of geodesic distances between adjacent cites in the sequence.

The planned routes for each salesman are post-processed by inexpensive 2-opt procedure that improves the solution without a great affect to the solution costs. The post-processing eliminates possible self-crossing of the found tours that can occur when dropping into a local optima due to distance among obstacles.

Neuron oscillations during the adaptation procedure were observed in some iterations for environments with obstacles. This oscillations occur in situations if neurons are not distributed uniformly among cities during the adaptation process. This leads to situations that one neuron is closest to two cities. The oscillations can be prevented by adjustment of a learning loop constrains - the learning procedure can be terminated if G is small enough. Another approach could be based on detection of oscillations of particular neuron between cities followed by adding neurons to the ring what eliminates oscillations.

4 Hierarchical adaptation process

If a visibility distance is small compared to the size of a working environment a number of sensing locations is too high to solve the MTSP problem in a reasonable time. Experiments show that a learning procedure for the neural network described in section 3 is achieved in less than 40 steps for parameters : $\alpha = 0.2, \mu = 0.6$ and initial value of $G = 12 \cdot n$, where n is number of cities. The value of G is high in the first cycles of the learning process inducing that neurons are moved by a long distances. During adaptation G is decreasing, what causes only local changes in the network. Finally, the learning is terminated whenever the longest distance from the closest neuron to each city is less than shortest distance between two cities. That means no other improvements can be made, while all cities has the unique closest neuron and movements of neurons are very small according to small value of G.
The time of the adaptation procedure depends on number of neurons that increases with a number of cities according to formula $M = \frac{2.5n}{m}$, where n is the number of cities and m is the number of salesmen. The number of cities (sensing locations) moreover increases with decreasing visibility distance.
The main idea behind speed improvement of the neural network for a large number of sensing locations is to train the network for a small number of cities representing the whole set of sensing locations in order to roughly sketch the paths. The number of cities is increased in the following steps so that a global shape of the paths stays unchanged while local parameters of the paths are modified in order to visit newly added cities.
To use this idea, a hierarchy of sensing locations (cities) is such that each layer of the hierarchy contains sensing locations for particular visibility distance is created. The construction of the set benefits from the fact that the algorithm for finding sensing locations is incremental. The hierarchical set can be therefore found as follows. Let n_1 be a number of sensing locations C_1 for a visibility distance d_1 and a visibility distance d_2 is smaller than d_1. Sensing locations for d_2 can be found as sensing locations that cover whole working space without area covered by locations in C_1 with the visibility distance d_2.
The set $\mathcal{C} = \{C_1, C_2, \ldots, C_h\}$ forms a hierarchy of cities. The neural network starts to learn positions of cities for the first layer of the hierarchy. This can be done very fast, while the number of cities is relatively small at this point. After that, the set of cities is extended with cities from the second layer. A number of neurons is now higher and the rings have to be recreated. The neurons that are closest to some city in the old ring are transfered to a new ring. Other neurons are equally placed between old neurons which lie on the cities and the adaptation process is re-started with increased G. The process is repeated unless the last layer of the hierarchy is achieved. This is shown in the Algorithm 1.

5 Experiments

Two polygonal maps were used in experimentally verify the developed algorithm - *TestArea* with 105 vertices and 2 holes and *PhysicalBuilding* with 105 vertices and 3 holes. The first map represents an office building with a large free-space, while the second one consists mainly of corridors. Sets of cities in hierarchies were computed for several visibility distances. Main parameters of hierarchies of cities is shown in Table 1. h denotes the number of layers in each hierarchy and the *Abbreviation* column contains a shortening that is used in consequent tables.

Algorithm 1: Self-Organizing Neural Network with Hierarchy of Cities. The hierarchy of cities denoted as the set $\mathcal{C} = \{\boldsymbol{C_1}, \boldsymbol{C_2}, \ldots, \boldsymbol{C_h}\}$, where $\boldsymbol{C_i}$ is set of cities such that $\boldsymbol{C_1} \subset \boldsymbol{C_2} \ldots \subset \boldsymbol{C_h}$. Number of neurons in the ring for the set of cities C_i is M_i.

Initial learning on $\boldsymbol{C_1}$, *for* $\delta = \delta_1$, $G = 12 \cdot |\boldsymbol{C_1}|$
foreach $\boldsymbol{C_i} \in \{\boldsymbol{C_2}, \ldots, \boldsymbol{C_h}\}$ **do**
    Extension of rings
        $\boldsymbol{N_c} = \emptyset$
        foreach $c \in \boldsymbol{C_{i-1}}$ **do**
            $\nu_c = argmin_{\nu \in N} \{distance(\nu, c)\}$
            $\boldsymbol{N_c} = \boldsymbol{N_c} \cup \{\nu_c\}$
        end
        foreach $r_i \in \boldsymbol{R}$ **do**
            $r'_i = \{\nu | \nu \in r_i \cap N_c\}$
            Add $\frac{|M_i| - |r'_i|}{|r'_i|}$ neurons equally between each two neurons in r'_i
        end
    $G := 2\frac{|M_i|}{|M_{i-1}|}$
    Learning on C_i *with rings* $r'_1, \ldots, r'_m$

Name	**Abbreviation**	h	**Visibility dist. [m]**	**Cities in hierarchy**
TestArea-258	T-258	3	4.5, 2.5, 1.9	56, 133, 258
TestArea-571	T-571	4	4.5, 2.5, 1.9, 1.3	65, 127, 246, 571
TestArea-1054	T-1054	4	5.4, 2.0, 1.2, 0.9	46, 215, 582, 1054
PhysicalBuilding-225	P-225	2	6.0, 3.0	84, 225
PhysicalBuilding-548	P-548	3	6.0, 3.0, 1.9	84, 229, 548

Table 1: Environment.

Several tests were performed for each hierarchy and for 2 and 3 salesmen. The maximal route length from each particular solution of MTSP was used to compute average and standard deviation values. All tests were performed on Pentium IV at 3.2GHz running Linux 2.6.9 and Java BlackDown 1.4.1. Computational results are shown in Table 2. The *Time* column show computational time in seconds, while *Time %* shows percentage of time of the proposed algorithm compared to time of the algorithm without the hierarchy of cities. *RL* describes maximal route length of the proposed algorithm over the algorithm without the cities hierarchy in percent. The longest route found by the proposed algorithm was not longer than 5% and time requirement was up to 37% of the algorithm without the cities hierarchy.

The highest number of iteration of learning procedure is in the first layer, Table 3. The column **Other** shows an average number of iteration in the next layers. **TTL** shows percentage of time spent in the last layer of the cities hierarchy. However the number of iterations in the last layer is smaller than in the first layer, major time is spent in the last layer.

The solution for the first layer of *TestArea258* is shown in Figure 1, number of cities is 56. The solution for the next layer, 133 cities, is shown in Figure 2. Final solution for 258 cities is shown in Figure 3.

Map	**n**	NN			Proposed				
		Time	**Avg**	**Dev**	**Time**	**Avg**	**Dev**	**Time %**	**RL %**
T-258	2	32.5	254.18	10.30	8.6	264.15	15.93	26.4	103.9
	3	29.9	181.42	6.84	7.8	188.90	8.37	26.2	104.1
T-571	2	185.7	342.37	7.71	48.2	354.36	13.06	26.0	103.5
	3	175.4	240.41	8.42	45.7	250.96	12.10	26.0	104.4
T-1054	2	699.5	441.21	6.23	165.5	456.64	18.67	23.7	103.5
	3	635.2	308.43	6.95	148.4	322.81	10.82	23.4	104.7
P-225	2	21.3	487.73	30.00	8.2	508.68	18.59	38.4	104.3
	3	20.7	408.16	33.68	8.1	408.55	28.50	38.8	100.1
P-548	2	142.6	594.23	32.17	37.8	607.43	21.34	26.5	102.2
	3	142.3	493.80	38.60	41.1	482.37	39.18	28.9	97.7

Table 2: Computational results

Map	**n**	**Iteration in Layers**			**TLL %**
		Total	**First**	**Other**	
TestArea-258	2	40	28	6	64.3
TestArea-571	2	50	30	6	82.3
TestArea-1054	2	49	27	7	77.6
PhysicalBuilding-548	2	42	29	6	83.1
PhysicalBuilding-225	2	40	32	8	67.1

Table 3: Number iteration and time in layers of hierarchy.

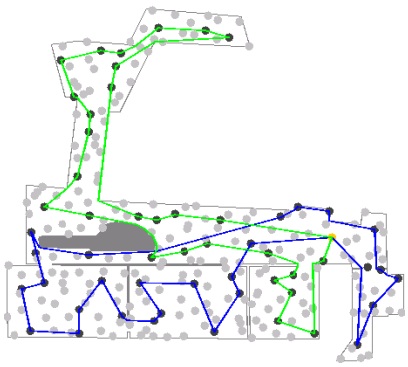

Figure 1: Learned net for first layer of hierarchy, TestArea.

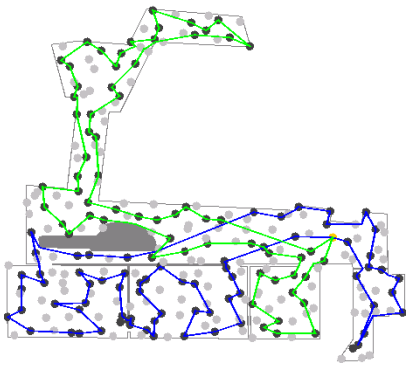

Figure 2: Learned net for second layer of hierarchy, TestArea.

Due to lower value of G in each new layer, a solution is dropped to a local optima. If the value of G is increased the neural network can leave the local optima. The influence of the new G value for an upper layer in a hierarchy was studied. A new variable `LGAIN` was introduced in step 11 of the Algorithm 1. `LGAIN` represents a *learning gain* parameter of the neural network in a new layer of the hierarchy of cities. Whenever value is increased, movements of neurons during adaptation are greater, i.e. a smaller amount of learned information from previous

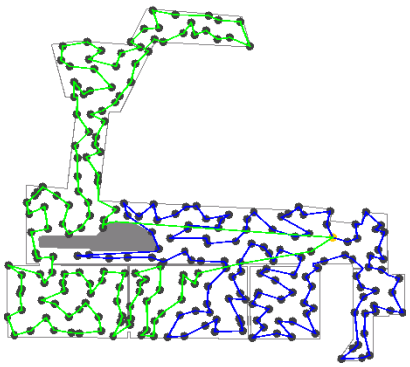

Figure 3: Learned net for last layer of hierarchy, TestArea.

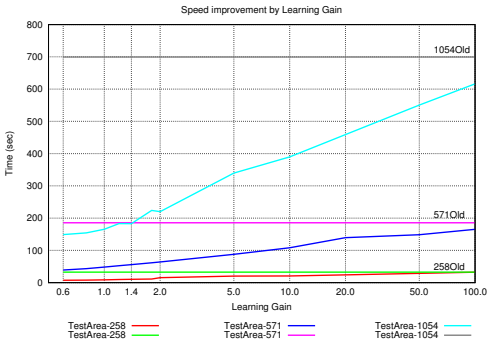

Figure 4: Time dependency on learning gain, 2 Salesmen, TestArea.

steps is used in the current step. In other words, the `LGAIN` variable defines the distance in which neighboring cities can move from their positions.
The assignment of G was modified to $G := \texttt{LGAIN} \cdot 2\frac{|M_i|}{|M_{i-1}|}$. Solutions of problem were found for several values of `LGAIN`. A dependency of solution time on the value of `LGAIN` for 2 salesmen is shown in Figure 4 and in Table 4.

Map	**n**	`LGAIN: 0.6`		`LGAIN: 1.0`		`LGAIN: 5.0`		`LGAIN: 20.0`	
		TM %	**RL %**	**TM %**	**RL %**	**TM %**	**RL %**	**TM %**	**RL %**
T-258	2	22.9	103.5	26.4	103.9	62.6	98.5	74.3	99.6
T-571	2	21.1	104.4	26.0	103.5	47.3	98.6	75.1	99.5
T-1054	2	21.3	104.6	23.7	103.5	48.6	100.7	65.6	99.9
P-548	2	23.1	102.5	26.5	102.2	45.5	100.6	64.9	100.5
P-225	2	35.4	105.1	38.4	104.3	57.7	103.1	74.9	101.3

Table 4: LGAIN influence.

6 Conclusion

An extension of self-organizing neural network approach of MTSP for large number of cities has been proposed. It uses hierarchical sets of cities, which are generated by a randomized dual sampling algorithm for AGP. A high number of cities arises from limitation on visibility distance of robots that performs an inspection task. The neural network is trained on a smaller problem. Then a learned structure of the environment is used to enforce learning procedure of a larger problem. This approach reduces the number of iterations in the largest problem, whereas time requirement of single iteration is the most expensive.
However, time complexity of the proposed algorithm in the last layer of the hierarchy is the same as time complexity of the algorithm without the cities hierarchy, the real time improvement of the proposed extension for learning procedure is approximately by factor 3. A cost of speed improvement is worse quality of solution, approximately by 4%. The speed improvement and quality of the solution can be controlled by the `LGAIN` parameter. Principally,

higher value of the parameter leads to a better solution but with higher time requirements. The original Somhom's algorithm was compared with heuristics, elastic net and adaptive tabu search in [3]. A comparison of the proposed approach without cities hierarchy with Ant Colony optimization and Genetic Algorithm was published in [4]. A quality of solutions were equal while the solution cost of the proposed approach is lower. The approach was tested in two environments for various number of cities. The environments were represented as a polygon with holes and they have been derived from a CAD model of a real building. One of the future work can be a testing proposed approach on other environments to get more significant results.
When visibility distance is very small the shortest distance between cities can be also very small. This leads to more learning steps. The learning procedure can be terminated before all the cities have a unique neuron. Than missing cities in planned routes have to be added into some route.
Additional speed improvement can be based on more local search procedure. Parts of a ring can be defined for each ring from previous learning procedure. In case we define for each city its neighborhood (i.e. cities that are not far than some certain distance) the adaption process can be local for each part of rings and neighborhoods of cities.

References

[1] M. Kulich, J. Kout, L. Přeučil, J. Pavlíček, and R. M. et al (2004), PeLoTe - a heterogenous telematics system for cooperative search and rescue missions, in *International Conference on Intelligent Robots and Systems*, **vol. 1**, Sendai, Tohoku Univeristy.

[2] H. González-Baños and J.-C. Latombe (1998), Planning robot motions for range-image acquisition and automatic 3d model construction, in *AAAI Fall Symposium*.

[3] S. Somhom, A. Modares, and T. Enkawa (1999), Competition-based neural network for the multiple travelling salesmen problem with minmax objective, *Computers and Operations Research*, **vol. 26**, no. 4, pp. 395–407.

[4] M. Kulich, J. Faigl, J. Kléma, and J. Kubalík 2004, Rescue operation planning by soft computing techniques, in *IEEE 4th International Conference on Intelligent Systems Design and Application*, Piscataway: IEEE, pp. 103–108.

[5] M. Bellmore and S. Hong (1974), Transformation of multisalesman problem to the standard traveling salesman problem," *Journal of the ACM*, **vol. 21**, no. 3, pp. 500–504.

[6] P. M. França, M. Gendreau, G. Laporte, and F. M. Müller (1995), The m-traveling salesman problem with minmax objective, *Transportation Science*, **vol. 29**, no. 3, pp. 267–275.

[7] T. Danner and L. Kavraki (2000), Randomized planning for short inspection paths, in *Proceedings of The IEEE International Conference on Robotics and Automation (ICRA)*, San Fransisco, CA: IEEE Press, pp. 971–976.

[8] J. S. B. Mitchell (1993), Shortest paths among obstacles in the plane, in *SCG '93: Proceedings of the ninth annual symposium on Computational geometry*, San Diego, California, United States: ACM Press, pp. 308–317.