Proceedings of the 2005 IEEE
International Workshop on Safety, Security and Rescue Robotics
Kobe, Japan, June 2005

Cooperative Planning for Heterogeneous Teams in Rescue Operations

Miroslav Kulich
Center of Applied Cybernetics
Czech Technical University
Technicka 2, 166 27 Prague 6
Email: kulich@labe.felk.cvut.cz

Jan Faigl, Libor Přeučil
Gerstner Laboratory
Czech Technical University
Technicka 2, 166 27 Prague 6
Email: {faigl,preucil}@labe.felk.cvut.cz

Abstract— **The first-place issue in cooperative activities of multiple entities in a common working environment is coordination and planning of systems activities ensuring completion of a common goal. As manual management of the problem is hard, finding a proper solution of the both tasks definitely leads to an efficient target behavior of these entities. In particular, the efficiency measured in terms of e.g. mission time, precise fulfilling of the common goal stands for the core issues in applications of a rescue type of scenarios and alike. The there under presented approach introduces a novel approach suitable for computer-aided or fully automatic mission planning and control within heterogenous teams of multiple robots and humans. The achieved performance of the approach has been experimentally verified and some interesting results are shown.**

I. Introduction

Inspection and exploration of an operating environment belong to the most interesting problems of robot path planning. There are mentioned many variations of this problem in the literature, differing in a kind of the environment (static x changing in time), environment map availability and representation, and a setup of the planning system (centralized x decentralized systems). When a map of the environment is not known, exploration is often solved together with mapping and localization tasks. Several algorithms based on frontier-based exploration were proposed. This work has been also extended to multi-robot teams [1].

Exploration of a known environment is defined as searching for an object in the environment. Two common environment representations are used: occupancy grid based maps and geometric maps. Computation geometry approaches are widely applicable for geometrical maps, while classical planning techniques based on A* or D* algorithms can be used for optimal planning on grid maps.

The classical *Watchman Route Problem* (WRP) deals with finding minimal closed route such that every point of the environment can be seen from at least one point along the route. Considering multiple watchmen operating in the same workspace (*Multiple Watchmen Routes Problem*), there are two measures evaluating found solutions. In case of the *MinSum* criterion, the aim is to minimize the sum length of watchmen routes, while the length of the longest watchman route is to be minimized for *MinMax* criterion. Nilsson [2] proved that both problems are NP-hard even for a simple polygon. A different (and more complicated) situation arises in case that the object to be found is moving. This problem is in the literature referenced as the P*ursuit-Evasion Problem*.

A complete exploration of a known environment can be done by visiting a set of points such that every point of the environment is visible from at least one such point. If the set of points is determined, planning can be defined as finding an optimal sequence in which to travel through all these points optimally. The WR problem is then divided into two problems that can be solved separately. This approach was described by Danner and Kavraki in [3].
The problem how to find an appropriate (i.e. minimal) set of sensing locations is called the *Art Gallery Problem*. The problem was posed by Victor Klee in 1973 who asked a question: "What is the smallest number of guards needed to guard an art gallery?" However a number of sufficient guards is proved, necessary number of the guards are lower in most cases.

A number of modifications and extensions of the classical *Art Gallery Problem* has appeared in literature. A survey can be found in [4]. The objective of the *Zoo-Keeper's Problem* and *Safari Route Problem* is to walk within a defined polygon containing sites (polygons) to be visited. While one can go thru the sites in the *Safari Route Problem* (like visiting pavilions at an exhibition), entrance to the sites is not allowed in the *Zoo-Keeper's Problem* (as a zoo-keeper feeding animals without going to their cages). These problems, which are NP-hard in general, were proposed by Chin and Ntafos in [5] have a tight relation to the problem of inspecting known environments such as office buildings. A cage can be replaced by a room in office building inspection in this case.

The previously mentioned approaches suppose unlimited visibility. This is not true in real applications because of bounded operational range of sensors. Moreover, the angle between the line of sight and the normal to the obstacle edge is also often limited. The limit of visibility is in the sense of sufficient detail objects. An approximate solution for randomized sensor placement with limited visibility was developed in [6].

The problem of planning optimal tour over found sensing locations could be defined as the classical T*raveling Salesman Problem* (TSP) task known in the operation research. The problem is defined on a graph *G(V,E)*, where *V* is set of vertices and *E* is set of connection over vertices with cost. A lot of heuristics finding solution close to the optimum have been proposed [7]. While the most successful approaches are based on the 2-opt heuristics, a recent research is focused to develop a meta-heuristics over these heuristics [8]. Theses approaches include but are not restricted Ant Colony Optimization, Genetic Algorithms, Simulated Annealing, etc. Moreover, several Neural Networks (NN) approaches have been applied to solve TSP. The three main approaches are *Hopfield Nets*, *Elastic Nets*, and *Self Organized Feature Maps* (SOM). The most promising approaches are based on SOM [9].

TSP defined for a team of multiple entities (robots) is called the *Multiple Traveling Salesmen Problem* (MTSP). This problem can be easily transformed to TSP, but unfortunately the solution is highly degenerated. Moreover, this transformation is usable only for *MinSum* criterion. From the other point of view, the MTSP can be also viewed as an instance of the *Vehicle Routing Problem* (VRP) in case when capacity is unlimited. VRP is described as follows: given a fleet of vehicles with uniform capacity and several costumer demands (represented as a collection of geographical scattered points), find the set of routes with overall minimum route cost which service all the demands.

II. Problem definition

The aim of the Search and Rescue scenario is to provide support for cooperating of heterogeneous teams (consisting of human and robots) for search and rescue activities in case of emergencies or catastrophes. The rescue squad of human and robots is operating in a large space with highly complex structure (building split into many small offices, storages, hallways, etc.). Both kinds of entities are expected to operate and navigate in the environment, identify properties of this area or even to create a model of the environment, to allow first aid services and rescue persons in non-standard critical situations.

Moreover, the diverse nature of the robot and human entities brings up multiple kinds of constrains for each of the type. These are mainly reflecting their operational capabilities and abilities with respect to the environment, like unsafe regions due to structural collapses, poisonous substances, high temperature and other reasons causing inaccessibility of particular spaces by a certain type of entity. The sketched additional constrains clearly form needs for smooth integration of possible constrains into the planning mechanism.

Basically, the core geometric information about the environment shape and structure can be carried in a map structure, common to all types of the entities. Then, the specific constrains are stored in a separate data -layer and being adjusted to operational properties of each particular entity

The problem of searching for objects in an area (or exploration of a workspace) by one entity can be defined as the *Watchman Route Problem*: given a known workspace and an entity with vision capabilities find a shortest closed path for the entity such that the path starts at the starting point *s* and ends at the goal point *g* and each point of the workspace is visible from some point on the path. While a typical rescue team consists of multiple members, the goal is not to find only path for one entity, but to plan a path for each entity separately, so that the found paths fulfill the *MinMax* criterion and each point of the workspace is visible from some point on at least one path.

Theoretical results show that no polynomial algorithm exists for this problem. Due to this, algorithms that don't generate an optimal (best) solution but find some feasible and good enough solution will be used. The algorithm described in the following chapters divides the Multiple Watchmen Route Problem into two problems that can be solved separately. This approach was used by Danner and Kavraki in [3] and is based on observation that sensing is a time consuming operation in real-world systems and that a detailed analysis and processing of data gathered from sensors is required. Therefore, it seams to be feasible to identify points in which the sensing will be performed. The two steps of the algorithm are following:

- Find a set of "sensing" locations (guards). This problem is called the Art Gallery Problem and it will be discussed in chapter III.
- Connect the found guards with *m* paths in an optimal way to form inspection paths. The problem can be formulated as the Multiple Traveling Salesmen Problem and it will be discussed in the chapter IV.

III. ART Gallery Problem

In order to get better than theoretically induced results and to deal with limited visibility of sensors used, several algorithms finding approximate solution of the problem were developed. We have implemented a randomised incremental algorithm described in [3] which is based on the approach introduced by González-Banos and Latombe [6]. The algorithm proceeds as a loop:

1. Denote *A* an area to be guarded.
2. A random point *p* lying on the border of the area *A* is chosen.
3. A polygon V_p is found, which consists of points visible from the point *p* (this is equivalent to the polygon from which *p* is visible). All the visibility constrains as defined above are applied.
4. *k* random samples p_k are placed into the polygon $V_{p.}$
5. For each point p_k a visibility polygon (polygon from which p_k is visible) is determined.

6. The guard (point) that can see the most still unguarded area (i.e. the point for which $\left|A - V_{p_k}\right|$ is

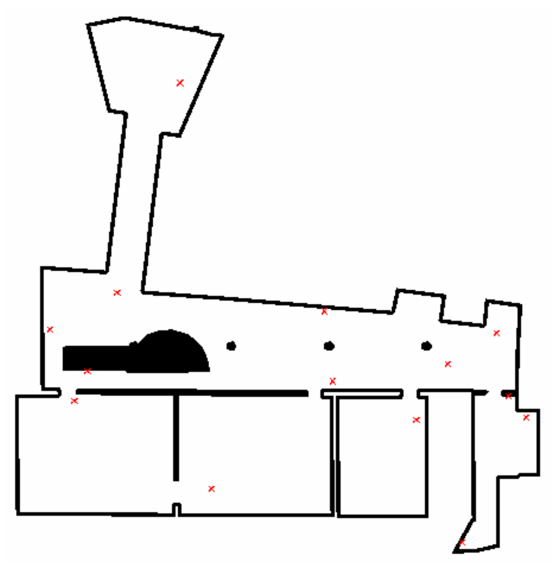

Fig. 1. Guards with an unlimited visibility range generated by the randomized incremental approach.

smallest) is chosen as a next guard.
7. Set $A = A - V_{p_k}$.
8. If A is not empty (there exist a point which is not guarded) then go to step 2).

IV. Multiple Traveling Salesmen Problem

Neural networks, and competition-based structures especially, are widely used for solving optimization problems. Our approach extends Somhom's algorithm described in [9]. The idea of the algorithm is to represent a path of each particular robot by a chain of neurons, where neighboring neurons are connected. At each iteration, a nearest neuron to a randomly selected guard is determined and together with its' neighbors moved closer to the guard. The algorithm can be described as follows:

If we denote n as the number of guards (cities) and m as the number of salesmen, then m chains can be created so that each consists of $M=2n/m$ neurons. Initially, the on-chain neurons are positioned on a small ring close to a starting point (depot) for each salesman.

The next steps then choose random permutations of the guards and existing neuron chains, i.e. C_{p_i} is the i-th city in a permutation, for which the nearest neuron to the C_{p_i} is determined. To select the nearest neuron, the guard-to-neuron distance is defined as their Euclidean distance weighted by: $weight(r)=((length(r)-AVG)/AVG)^4$. This suppresses those neurons, which overshoot the AVG and prefers those, which are bellow the *AVG*, where the AVG stands for the average length of chains in the task.

Whenever determining the minimum distance neuron, the winner and its' nearest neighbors on the chain are attracted to the guard. The value of the winner movement towards the guard is proportional to its' distance to the guard and additionally weighted by exponential function of the distance and the iteration number. This ensures rapid changes in the network topology for larger neuron-to-guard distances and more precise and slow convergence at final phases of the iteration process.

The process of the permutation choice, nearest neuron selection and shifting towards proper guard is executed unless a termination criterion is achieved. This can be defined as a maximum distance between a guard and a nearest neuron to this guard being satisfied if the distance is smaller than a certain threshold.

The above-described algorithm gives results comparable to other soft-computing techniques: genetic algorithms and ant colonies optimization but works much faster than these approaches. Unfortunately, the algorithm works only in the case that distances between a neuron and a guard can be determined effectively. It is true for environments without obstacles, where the distance is clearly an Euclidean distance.

The situation is more difficult for environments with obstacles (represented by a polygon with holes), because the distance between two points has to be determined with respect to these obstacles. The simplest approach is based on visibility graphs:

1. Let C is a city and $\{V_i\}$ for $i=1..N_v$ is a set of vertices of obstacles.
2. Construct a visibility graph of $\{C\}\cap\{V_i\}$.
3. Use Dijkstra algorithm to compute a shortest path and its length from C to each vertex V_i. Denote the i-th path P_i and its length d(CV_i).

Using this structure, queries "what is the distance of a point p to C" (so-called one-point queries) can be answered in *O(nlogn)* time:

1. Determine vertices that are visible from point p and denote them $V^p = \{V_i^p\}$, where i=$1..N_p$. Furthermore, $|v_i,p|$ stands for the Euclidean distance of v_i and p.
2. Find a vertex $v \in V^p$ so that the length of from C to v plus the length from v to p is minimal over V^p:

$$v = \arg\min_{v_i \in V^p} d(C, v_i) + |v_i p|. \qquad (1)$$

3. The desired minimal path from C to p is then the union of the path from C to v and the line vp while the distance is $d(C,v)+|vp|$. The distance of two points with respect to obstacles is called Euclidean geodesic.

O(nlogn) time of this naive algorithm is to high what causes that the neural net optimisation runs several minutes even for a scene with 250 vertices. Therefore more sophisticated structure has to be used that provides answering one-point queries more effectively.

The structure we use is the Shortest Path Map (SPM). SPM(C,*map*) with respect to point (city) C and polygon with holes *map* is a partition of a free space defined by *map* into maximal regions (called cells) that correspond to sets of points with the same root (first vertex on the shortest path from a point to *C*, i.e. *v* in the naïve algorithm) or a set of roots with respect to *C*.

We use a subquadratic algorithm for computing the SPM presented in [10]. This approach is based on continuous Dijsktra paradigm, which simulates the effect of a "wavefront" propagating out from a city. The wavefront at a distance *d* is the set of all points of free space that have an Euclidean geodesic distance equal to *d*. The structure of the wave-front changes while *d* is growing only at some specific distances – events due to the following three possibilities:

- a part of the wavelet disappears,
- the wavelet collides with an obstacle vertex, or
- a part of the wavelet collides with other part.

The events correspond to changes in the SMP. The basic idea of the algorithm is therefore to detect these events and to process them.

The SMP is a subdivision of a plane, where boundaries between two cells can be either a portion of straight line or a hyperbolic arc. If each hyperbolic arc is approximated by a polyline (with a defined precision) a cell can be represented by a polygon. Having a SMP for each city, we can compute the overlay of these subdivisions using standard plane sweep algorithm [11] sequentially:

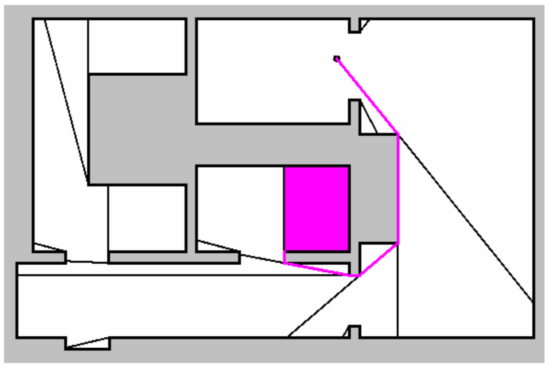

Fig. 2. The Shortest Path Map structure generated for the black point. Points from the highlighted area have the same shortest path to the point.

1. Let SMP_i is a shortest path map with respect to the city C_i, $i=1..N$ and *S* stands for a resulting subdivision.
2. $S = C_1$.
3. For $i= 2..N$ do $S = S \cap C_i$, where $\cap$ stands for the overlay.

After applying the previous algorithm, we get a subdivision with the following properties:

- Each cell is represented by a polygon
- For each city and cell there exists one root that is same for all points of the cell. In other words, all points of the cell have the same path topology to a city.

These properties guarantee that the subdivision allows to find the length of the shortest path between an arbitrary

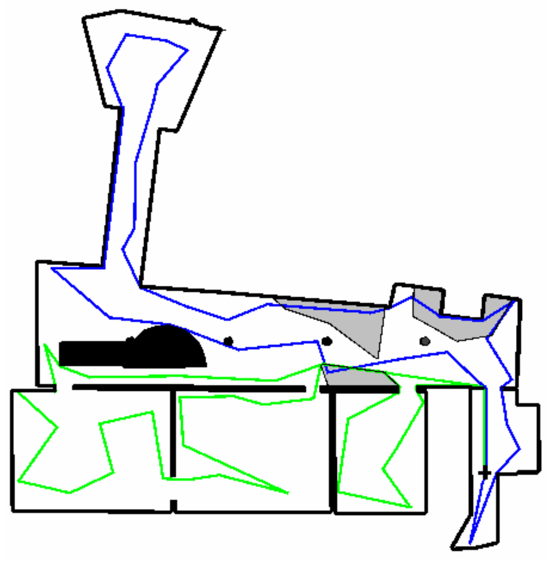

Fig. 3. Paths generated for two entities. The green entity is not allowed to go thru the grayed areas.

point of the plane (neuron) and a city in time *O(log k)* (by point location), where *k* is the number of points of the subdivision. Moreover, a shortest path can be produced in time *O(log k + b),* where *b* is the number of bends (vertices) in the path.

Additional improvement of computation time of finding a cell in which a neuron lies during a self-organization process of a neural net is based on fact, that a neuron is not placed randomly in the place but moves on the shortest path to some city only. A cell, where a neuron lies ca be then effectively determined by the following process:

1. Let $Cell_a$ is a cell where the neuron *p* is placed and its position is adapted so that it is tightened to the city *C*. Denote obstacle vertices lying on the shortest path from *p* to *C* as $v_1, v_2, \ldots, v_d$.
2. Compute the distance at which the neuron will travel during the adaptation step similarly to step 3b) in the original algorithm:
$$L = \mu f(b,k) * d(C,p), \qquad (2)$$
where d(C,p) is the length of the shortest path between *C* and *p* (i.e. their geodetic distance) and μ and *f* have the same meaning as in the original algorithm.
3. Determine the point *q* that lies on the shortest path between *C* and *p* and its geodetic distance is equal to *L*. Denote the portion of the path on which *q* lies $v_x v_{x+1}$.
4. Determine the cell in which the point v_x lies and denote it $Cell_x$. This information can be obtained during SMP-building process and stored for each obstacle vertex.
5. Detect whether *q* lies in $Cell_x$. If yes then stop - $Cell_x$ is the desired cell. Otherwise determine intersection $Cell_x \cup v_x q$. Denote the intersection point v_x. The new point v_x lies on the boundary of two cells - $Cell_x$

and one of its neighbors. Denote this neighbor $Cell_x$ and go back to step 5.

The time complexity of steps 4 and 5 is *O(m)*, where m is the number of cell vertices. Moreover, step 5 is repeated as many times as many cells are visited on the path between *p* and *q* – denote this number *r*. The complexity of step 2 is

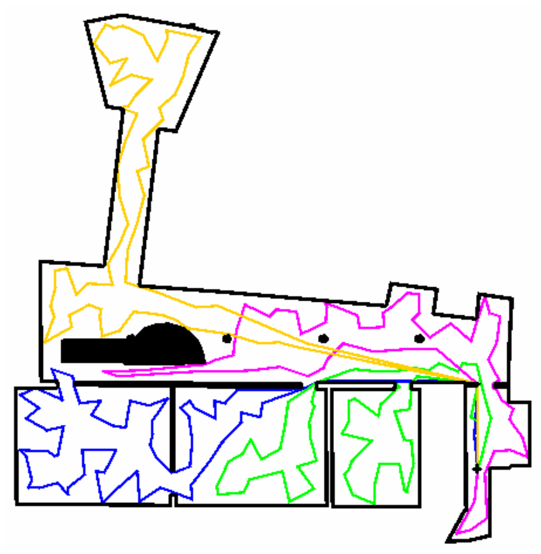

Fig. 4. Paths generated for four entities having a small visibility range.

constant while the complexity of step 3 is *r*. Therefore the overall complexity of the algorithm is *O(mr)*. On the assumption that the distance between *p* and *q* is small (which is a frequent case in the self-organizing process) *r* is constant and therefore the complexity of the algorithm is *O(m)*.

The second aspect to be taken into account is computation of the equation in step 3a) of the original algorithm. In order to evaluate this equation, lengths of rings have to be determined, which leads to enumeration of distances between neurons. Unfortunately, determination of neuron-neuron distances is very time consuming, because no structure like SMP for neuron-city distances can be used. Therefore, instead of computing lengths of rings directly, we compute the length of a path travelled by each salesman in the following procedure:

1. For each city C_i determine a nearest neuron n_i..
2. For each ring *r* do

a) Go neuron by neuron as they are connected and for each neuron, which is nearest to some city, remember this city. By this process we get an ordered list of cities $C_{r_1}, C_{r_2}, \dots, C_{r_n}$.

b) Compute the length of the ring as a sum of geodesic distances between connected cities in the list:

$$length(r) = \sum_{i=1}^{n-1} d(C_{r_i}, C_{r_{i+1}}). \qquad (3)$$

In order to compare quality of the right described approach we performed several tests (20 or 100) in three different environments and for 2,3, and 4 salesmen. Each test was performed for the original method based on neuron-to-neuron distance and for the method described in this chapter. After performing the tests we evaluated quality of each found solution using *MinMax* criterion and computed minimal, maximal, and average number over all test with the same configuration. The results (see Table 1) show that both approaches give at least comparable results. In majority of tests the city–to-city distance is better which qualifies us to use this method.

Finally, found paths of particular entities are optimized by 2-opt heuristics, which is commonly used in TSP-like tasks. The heuristics generates so-called 2-optimal tour, i.e.

TABLE I
Comparison of approaches for determination of the length of a ring. Method "neuron" computes the lengths as used in original algorithm, while "city" determines the length using distances between cities.

Map	**Nbr. of salesmen**	**Method**	**Nbr. of tests**	**Path Length**		
				min	**max**	**avg**
1	2	city	100	8997.17	10887.93	9645.0
		neuron	100	8997.17	14826.36	9696.1
	3	city	100	6958.39	9270.25	7722.4
		neuron	100	7059.09	10304.31	8037.1
	4	city	100	6471.52	8552.99	6959.9
		neuron	100	6549.30	9528.09	7702.4
2	2	city	20	1463.93	1625.87	1522.8
		neuron	20	1469.19	1878.08	1594.6
	3	city	20	1115.66	1448.10	1226.6
		neuron	20	1113.08	1462.76	1255.5
3	2	city	100	1767.39	2431.58	2002.9
		neuron	100	1767.38	2235.61	1948.1
	3	city	100	1346.20	2050.81	1606.8
		neuron	100	1338.70	1966.26	1576.2
	4	city	100	1218.11	2041.07	1434.7
		neuron	100	1201.70	2051.82	1472.8

tour in which there is no possibility to shorten the tour by exchanging two arcs.

V. Conclusion

The contribution brings up the question of optimal activity planning and scheduling in cooperating teams of entities. The central novelty of the approach presented there above stands in investigation of a robust planning technology for heterogeneous teams consisting of robots and human actors, sharing a joint task in a common environment. Combining diverse types of entities takes the advantage of having complementary properties (or abilities) provided by the entities itself, what seems to be efficient for the task solution. On the other hand, substantially different entities have also diverse constrains in their capabilities, which have to be taken into account by the discussed approach. Moreover, the suggested method incorporates the planning constrains in a flexible way, that can be extended to other features as well as modified in time. The option of time-varying conditions supports the on-line re-planning, which allows modifying current plans to variations in the task setup or environment status.

The investigated method has been implemented and experimentally verified in a search and rescue scenario (a complete space inspection problem) as a part of the PeLoTe system (PeLoTe is a hybrid telematic system for cooperative rescue operations in cases of emergences or catastrophes [12][13][14][15][16]).

Although the core of the proposed approach relies on solution of the MTSP problem, which is an NP-complete task, the proposed approach offers close-to-optimum solution achievable in real-time for typical indoor environments with computational power of a standard PC.

The remaining simplification of the presented approach seems to stand in splitting the problem solution into two independent steps: the Art Gallery and MTSP problems solution. The expected future developments of the method are aimed at natural integration of these two parts of the problem. This direction seems to be promising with respect to further improvement of the method performance.

Acknowledgement

The work has been supported within the IST-2001-FET framework under project no. 38873 "PeLoTe Building Presence through Localization for Hybrid Telematic Systems". The support of the Ministry of Education of the Czech Republic, under the Project No. 1M6840770004 to Miroslav Kulich is also gratefully acknowledged.

References

[1] D. Fox, W. Burgard, M. Moors, R. Simmons, and S. Thrun, Collaborative multi-robot exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[2] B. J. Nilsson, *Guarding Art Galleries - Methods for Mobile Guards*. PhD thesis, Lund University, 1995.

[3] T. Danner and L. Kavraki, Randomized planning for short inspection paths. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 971–976, San Francisco, CA, April 2000.

[4] J. Urrutia, *Handbook of Computational Geometry*, chapter Art gallery and illumination problems, pages 973–1027. Elsevier Science Publishers, B.V. North Holland, Amsterdam, 2000.

[5] W. Chin and S. Ntafos, The Zookeeper Route Problem. *Inf. Sci.*, 63(3):245–259, 1992.

[6] J.C. Latombe and H. Gonzlez-Banos, A randomized art-gallery algorithm for sensor placement. In *Proceedings of the seventeenth annual symposium on Computational geometry*, pages 232–240. ACM Press, 2001.

[7] G. Gutin and A. Punnen, *The Traveling Salesman Problem and its Variations*. Kluwer Academic Publishers, Dordrecht, 2002.

[8] Ch. Blum and A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, 2003.

[9] S. Somhom, A. Modares, T. Enkawa. Competition-based neural network for the multiple travelling salesmen problem with minmax objective. *Computers & Operations Research*, 26(4):395–407, 1999.

[10] J. S. B. Mitchell: *Shortest paths among obstacles in the plane,* International Journal of Computational Geometry and Applications, 6(3), pp 309-332, 1996.

[11] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf.: *Computational Geometry: Algorithms and Applications,* Springer-Verlag, 2000.

[12] M. Kulich, J. Kout, L. Preucil, L. et al.: *PeLoTe – a Heterogeneous Telematic System for Cooperative Search and Rescue Missions*. Urban search and rescue: from RoboCup to real world applications, in conjunction with the 2004 IEEE/RSJ International conference on Intelligent Robots and Systems (IROS) Sendai (Japan), 2004.

[13] F. Driewer, K. Schilling, H. Baier,Human-Computer Interaction in the PeLoTe rescue system,IEEE International Workshop on Safety, Security, and Rescue Robotics (SSRR05), June 6-9, 2005, International Rescue System Institute, Kobe, Japan. To be appear in PeLoTe Session of SSRR05.

[14] N. Ruangpayoongsak, H. Roth, K. Schilling, J. Chudoba, Mobile robots for search and rescue, Workshop on Safety, Security, and Rescue Robotics (SSRR05), June 6-9, 2005, International Rescue System Institute, Kobe, Japan. To be appear in PeLoTe Session of SSRR05.

[15] R. Mazl,J. Pavlicek and Libor Preucil, Structures for Data Sharing in Hybrid Rescue Teams, IEEE International Workshop on Safety, Security, and Rescue Robotics (SSRR05), June 6-9, 2005, International Rescue System Institute, Kobe, Japan. To be appear in PeLoTe Session of SSRR05.

[16] J. Saarinen,,S. Heikkilä, M. Elomaa, J. Suomela and A. Halme, Rescue Personnel localization systém. IEEE International Workshop on Safety, Security, and Rescue Robotics (SSRR05), June 6-9, 2005, International Rescue System Institute, Kobe, Japan. To be appear in PeLoTe Session of SSRR05.