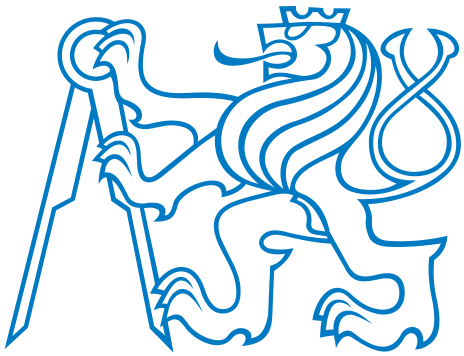
CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF ELECTRICAL ENGINEERING

Doctoral Thesis



February, 2010

Jan Faigl

CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF ELECTRICAL ENGINEERING
DEPARTMENT OF CYBERNETICS
GERSTNER LABORATORY

MULTI-GOAL PATH PLANNING FOR COOPERATIVE SENSING

Doctoral Thesis

Jan Faigl

Prague, February, 2010

Ph.D. Programme: Electrical Engineering and Information Technology
Branch of study: Artificial Intelligence and Biocybernetics

Supervisor: Ing. Libor Přeučil, CSc.

Abstract

The thesis deals with the multi-goal path planning problem. The problem is studied in the context of the inspection task of cooperating robots in a search and rescue mission. Two models of the sensing are considered: continuous and discrete. An environment is a priori known and it is represented by a polygonal domain in both approaches. The continuous sensing assumes relatively inexpensive cost of sensing in comparison to the cost of motion and can be formulated as the Watchman Route Problem (WRP). The discrete sensing leads to the decoupled approach that is motivated by problems where the cost of sensing is dominant. The decoupled approach consists of two problems: the problem of finding minimal set of sensing locations, and the multi-goal path planning problem to find a path visiting the found locations. The set of sensing locations can be found as a solution of the Art Gallery Problem (AGP) or sensor placement problem if additional visibility constraints have to be considered, e.g. visibility range, incident angle. The multi-goal path planning problem can be formulated as the Traveling Salesman Problem (TSP) in which paths between cities have to be traversable by the robot. All three problems (WRP, AGP, TSP) are known to be NP-hard, thus approximate algorithms are considered to find solutions in a reasonable time. The cooperation of several robots is formulated as the multi-robot variant of the WRP, resp. the TSP, with the MinMax criterion.

A new sensor placement algorithm, called the Boundary Placement is proposed to find a set of sensing locations with restricted visibility range. The algorithm is compared with approaches based on convex polygon partitioning and the randomized dual sampling schema. The proposed algorithm outperforms both algorithms in the number of found locations and also in the required length of the inspection path.

The Self-Organizing Map (SOM) for the TSP is considered in a polygonal domain $\mathcal{W}$ in order to solve the multi-goal path planning problem. The SOM adaptation procedure modifies weights of neurons according to city presented to the neural network. The weights represent nodes that are organized into a ring of nodes and the ring represents a solution of the TSP. Computations of node–city paths and distances are supported by three approaches: an approximation of the geodesic path based on the convex partition of $\mathcal{W}$, navigation functions, and a triangular mesh of $\mathcal{W}$. The proposed algorithm is computationally feasible and it provides competitive results to the GENIUS heuristic.

The SOM based algorithm with navigation functions is applied to the generalized multi-goal path planning problem where a goal can be represented by a set of points instead of a single point. The proposed algorithms are demonstrated in the inspection planning with segment sensing locations, and in the problem of cooperative visits of convex areas of interest.

A new adaptation procedure is proposed to solve the WRP with restricted visibility range in a polygonal domain $\mathcal{W}$. The computation of continuous sensing is supported by a triangular mesh of $\mathcal{W}$ and a convex cover of $\mathcal{W}$. The procedure is also used to find solutions of the MWRP variants: independent patrolling routes, and closed routes starting from a common depot.

The problem of the multi-goal path planning with respect to kinematic and kinodynamic constraints is considered as a trajectory generation based on the found path. Moreover a new motion planner called RRT–Path^{ext} is proposed. The planner is able to find a solution of the multi-goal motion planning problem and it is used to regenerate a ring of nodes in the SOM based algorithm for the WRP. A solution of the WRP is found as an inspection trajectory for one or several mobile robots satisfying kinematic and kinodynamic constraints.

"Any sufficiently advanced technology is indistinguishable from magic."

Arthur C. Clarke

Acknowledgments

I would like to express my thanks to my supervisor Libor Přeučil and colleagues at Intelligent and Mobile Robotics Group. I would like to express my special thanks to Miroslav Kulich who introduced me to the problematic and provided me valuable suggestions. My thanks also go to my student Jan Mačák for his implementation of the finite element method and Vojta Vonásek, who became my colleague at IMR, for his implementation of the GENIUS algorithm, and collaboration in motion planning. Lastly, I would like to thank my family for their endless toleration and support during my research.

The work presented in this thesis was supported by the European Union projects FP5-IST-38873 "PeLoTe", FP6-IST-506958 "ECOLEAD", Ministry of Education of the Czech Republic under projects Kontakt 10-2005-06, 2C06005, MSM6840770038, and FRVŠ grants G1 2042/2004, G1 1606/2005 and by Czech Technical University grant CTU 0505413.

Contents

Abbreviations

AGP	Art Gallery Problem
AOI	Area of Interest
APF	Artificial Potential Field
ART	Angle-Restricted Tour
CW	ClockWise
CCW	CounterClockWise
FEM	Finite Element Method
FPGA	Field-Programmable Gate Array
GIS	Geographical Information System
MST	Minimum Spanning Tree
MTSP	Multiple Traveling Salesmen Problem
MWRP	Multiple Watchmen Routes Problem
NN	Neural-Network
PRM	Probabilistic RoadMap planner
RRT	Rapidly-Exploring Random Tree planner
SOM	Self-Organizing Map
SLAM	Simultaneous Localization and Mapping
TSP	Traveling Salesman Problem
UAV	Unmanned Aeiral Vehicle
VPP	View Planning Problem
VRP	Vehicle Routing Problem
WRP	Watchman Route Problem

Glossary

Robot is a collection of bodies capable of generating their own motions.

Workspace, $\mathcal{W}$, is a subset of 2D physical space: $\mathcal{W} \subset \mathbb{R}^2$ in this thesis.

Configuration is any mathematical specification of the position and orientation of every body composing a robot, relative to a fixed coordinate system.

Configuration Space, C-Space or just $\mathcal{C}$, is a set of all configurations of a robot. For a rigid robot $\mathcal{A} \subset \mathcal{W}$ and an obstacle region $\mathcal{O} \subset \mathcal{W}$ expressed by polygonal models the *obstacle region* $\mathcal{C}_{obs} \subset \mathcal{C}$ is $\mathcal{C}_{obs} = \{q \in \mathcal{C} | \mathcal{A}(q) \cap \mathcal{O} \neq \emptyset\}$ and free space is denoted as $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$.

Polygonal Path is a continuous map of an interval from a point s to point t consisting of a finite number of line segments (edges) joining a sequence of points.

Path is a continuous map $\tau : [0, 1] -> \mathcal{C}_{free}$. In this thesis, a path is always polygonal path and even in a case it is found as continuous smooth function, it is always discretized into finite number of line segments.

Trajectory is a path parametrized by time, and velocities and accelerations can be computed by taking the first and second derivatives with respect to time.

Simple polygon P, having n vertices, is a closed, simply-connected region whose boundary is a union of n (straight) line segments (edges), whose endpoints are the vertices of P.

Polygonal domain P, having n vertices and h holes, is a closed, multiply connected region, whose boundary is a union of n lines segments, forming $h + 1$ closed (polygonal) cycles.

Obstacle is any region of the space whose interior is forbidden to paths.

Free space is complement of the set of obstacles. If the free space is a polygonal domain P, the obstacles are the $h + 1$ connected components (h holes and the face at infinity) of the complement of P.

Map is a model of robot surrounding environment, a polygonal domain representing free space is used to model the environment in this thesis.

Feasible path is a path such that all point of the path are reachable by the robot from its first point.

Shortest path is a path of minimum length among all paths that are feasible, satisfying all imposed constraints.

Geodesic path is the shortest path among obstacles.

Diagonal is a line segment connected two polygon nonadjacent vertices and contained in the polygon, an *edge* connects adjacent vertices.

Visibility - Two points p and q in a polygon P are called *visible* if the line segment joining them is contained in P.

***d*-Visibility** - Two points in a polygon P are called *d-visible*, if they are *visible* and the length of the line segment joining them is less or equal to d.

Internal angle - A vertex of simple polygon P defines two angles a non-internal angle and an internal angle. The angle that intersects P is called *internal angle*.

Reflex vertex - A vertex v of polygon P is a *reflex vertex*, if its internal angle is strictly greater than π. Otherwise the vertex is called *convex vertex*.

Steiner point - A point is called *Steiner point* if it is an extra vertex that is not a member of the input.

Monotone polygon is a polygon for which any line parallel to some fixed direction intersects the polygon in a single connected piece.

Convex polygon - A polygon P is called *convex* if the line segment joining any two points of P lies within P.

Star shaped polygon - A polygon P is called *star shaped* if there is a point $q \in P$ such that all points of P are visible from q.

Polygon cover is a collection of sub-polygons whose union is exactly the input polygon.

Polygon partition is a collection of sub-polygons within pairwise disjoint interiors whose union is exactly the input polygon.

Guard is a point that is source of visibility or illumination.

Sensing location is a guard with realistic physical visibility constraints.

Point visibility polygon is the region visible from a point in a polygon P.

Segment visibility polygon is the region visible from a segment in a polygon P.

Ray-shooting query is a query asking which object is first hit by a ray oriented in a given direction from a given point.

Path planning problem: Compute a free or semi-free path between two input configurations.

Motion planning problem: Find a feasible trajectory from a initial position (configuration) to the given destination.

Touring polygons problem: Find a shortest path/cycle that visits in the given order at least one point of each polygon in a sequence $(P_1, P_2, \ldots, P_k)$.

Monotone polygonal path - A path is called *monotone* if there exist a direction vector d such that every directed edge of the path has a non-negative inner product with d.

Monotone path problem: Find a shortest monotone path (if any) from s to t in a polygonal domain P.

Sailor's problem: Compute a minimum-cost path, where the cost of motion is direction-dependent, and there is a cost L per turn (in a polygonal path).

Watchman Route Problem (WRP): Find a shortest cycle (path) within a polygonal domain P such that every point of P is visible from some point of the cycle.

Lawnmowing problem: Find a shortest cycle (path) for the center of a disk such that every point of a given region is covered by the disk at some points along the cycle (path). The problem is closely related to the WRP with restricted visibility (d-sweeper).

Zookeeper's problem: Find a shortest cycle in a simple polygon P (the zoo) through a given vertex v such that the cycle visits every one of a set of k disjoint convex polygons (cages), each sharing an edge with P.

Safari route problem: Find a shortest tour visiting a set of convex polygonal cages attached to the inside wall of a simple polygon P.

Traveling Salesman Problem (TSP): Find a shortest cycle that visits every point of a set S of n points.

City is a point to be visited in the TSP, it can be found as guard or as sensing locations. The city is one of the goal in the multi-goal path planning formulated as the TSP.

Chapter 1

Introduction

Navigation is the fundamental problem of mobile robotics. It can be defined as a set of techniques that provide abilities to safely move a mobile robot to a desired location. The behaviour of the mobile robot acting in an environment can be called intelligent, if the robot respects the environment while it is navigated towards the goal and selected criterion of optimality is considered. Such desired capabilities of the intelligent mobile robot can be accomplished by a reasoning about the environment and actions to be performed. The reasoning capabilities are represented as the cognition planning module in the general architecture of the intelligent mobile robot shown in Figure 1.1.

The multi-goal path planning is one of the cognitive activity that aims to provide a path for a robot such that the robot visits given set of places (goals) while a selected criterion is optimized, e.g. the path is the shortest one. The set of goals can be given by a human operator, or can be found automatically as a part of the planning process to fulfill the given task. If several robots can participate on the task, two additional aspects can be considered: cooperation and coordination. The cooperation deals with efficient distribution of goals among participating robots, while the coordination deals with efficient sharing of common resources. The common resource is the workspace of mobile robots in the path planning problem, thus the coordination avoids possible collisions between robots.

Regarding the architecture of the intelligent robot, Figure 1.1, the planning is related to the model of the environment and to the plan (path) execution module. An environment representation can affect suitability of a particular planning algorithm. The plan execution module relates to particular robot parameters and it can define motion constraints that should be respected in the path planning. These relations have to be considered in the particular planning algorithm, because possible real application of such an algorithm on mobile robots is highly dependent on capabilities to create a suitable world model and feasibility of the planned path.

The related modules should be considered in the context of particular task, which have to be solved by mobile robots, thus the problem motivation of this thesis is presented in the next section.

1.1 Problem motivation

The main motivation for the studied multi-goal path planning problem is a *search and rescue mission* in an office like environments for a team of robotic and human entities.

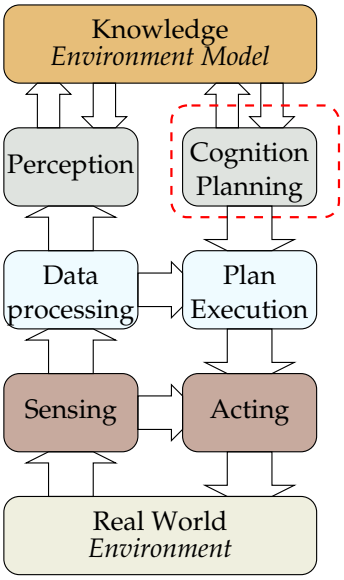


Figure 1.1: Schema of cognitive robot.

A request for a practical solution of the problem comes from the IST-2001 FET project number 38872 - PeLoTe - Building Presence through Localization for Hybrid Telematic Systems, which deals with cooperation of human and robotic entities in a search and rescue scenario [160]. The planning of cooperative motion is one of the key components of the whole system. The problem is an efficient searching in a priori known environment, where changes of the environment can occur. Despite the fact that an environment can be changed after a disaster or even it can be changed during the mission a preliminary planned paths are used at the mission start and they serve as guidelines how to search the environment to find possible victims as fast as possible. The time to complete the searching is the most relevant and in the case of multi-robotic team it leads to minimize the longest inspection tour.

In a mission scenario for an office like building a CAD model is a priori known and it is used by the rescue team leader to plan the rescue operation. Such model can be used as a model of the environment in a path planning algorithm to find a tour for each member of the team. Thus, the environment is known and the problem can be formulated as the inspection task:

> ***Inspection Task*** *- For a given group of mobile robots find a path for each robot such that all interesting parts of the environment will be seen by at least one robot.*

Two particular constraints are part of the task: paths must be traversable by the robot and have to respect sensing capabilities of the robot. In addition, an algorithm for the inspection task have to provide solution in a reasonable time to be used in practical experiments of real rescue mission.

1.2 Inspection Task

The inspection task deals with a problem to "see" all interesting parts of the environment, thus it can be categorized into the class of visibility problems. Two types of the sensing can be considered in the inspection task: continuous sensing and discrete sensing. The visibility problems are studied in the computational geometry, where a problem formulation

of the inspection task for the particular model of sensing can be found. The continuous sensing is motivated by problems where the cost of the sensing is relatively cheap in comparison to the cost of the motion. Contrary to the discrete sensing model, where the cost of the sensing is dominant and the cost of the motion can be ignored. The inspection task with the continuous sensing can be formulated as the Watchman Route Problem (WRP).

> ***Watchman Route Problem*** *- Find the shortest cycle (path) within a polygonal domain P such that every point of P is visible from some point of the cycle [109].*

The higher sensing cost can be caused by limited computational power carried on a robot, which does not allow continuous data processing, or measurement cannot be performed during the robot motion, because it requires a significant amount of time and vibrations can decrease quality of measurements. The discrete sensing leads to the decoupled approach to solve the inspection task. At first, a set (possibly minimal) of sensing locations is found such that the environment will be inspected by performing measurement at these locations. After that, a path to visit the found set is determined. The problem to find a set of sensing locations can be formulated as the Art Gallery Problem (AGP).

> ***Art Gallery Problem*** *- Find a minimal number of guards (sensing locations) within a polygonal domain P such that every point of P is visible from some guard.*

The problem of path planning to visit sensing locations is an instance of the multi-goal path planning problem, which satisfy mission objective of the inspection task. The multi-goal path planning problem can be formulated as the well know Traveling Salesman Problem (TSP).

> ***Traveling Salesman Problem*** *- Given a set of cities (goals) and distances between them, determine the shortest path starting from a given city, passing through all the other cities and returning to the first city.*

The route is closed in the TSP, which is not a drawback in the inspection task, because a rescuer typically enters to a building at the emergency exit, which is also used to leave the building.

The combination of the both costs is a difficult problem and it remains largely unexplored [107]. The decoupled approach has been successfully applied for the inspection planning in [69], therefore it is preferred in order to fulfill the problem motivation. The inspection task can be solved in four steps:

1. solve the AGP for the given environment to obtain a set of sensing location,
2. plan paths between each sensing locations,
3. solve the TSP to get a plan how to visit sensing locations,
4. use the planned path to guide the navigation of robot in the environment.

Both problems: the AGP and the TSP, are known to be NP-hard, therefore only approximate solution can be expected rather than optimal in a reasonable computational time. Moreover particular constraints related to the robotics have to be considered in the AGP and the TSP, which are not directly addressed by the common problem formulations. Sensing locations have to be reachable by the used mobile robots, also it is clear that paths between sensing locations have to be collision free and have to be traversable by the particular mobile robot.

The TSP formulation requires costs between cities. The cost is related to a path between cities, it can be the length of the path, or time to traverse the path. To determine required time to traverse the path a trajectory have to be generated. In such case, the optimal control of the robot movements along the trajectory can play an important role

in minimization of the required time and the problem becomes relevant to the control-theory. It is worth to mention that approaches from the control-theory tend to apply to systems with particular structure and no obstacles [52]. An optimization of a feasible path such that optimized path satisfies all constraints and its quality is improved is called trajectory optimization. The trajectory optimization problem is considered complementary to motion planning, because it usually requires an initial guess [176]. Therefore the thesis is concerned to path planning and length of the path is used as an estimation of the travel cost.

From the perspective discussed in the previous paragraphs, the inspection task can be decomposed into sequence of relatively independent problems, see Figure 1.2. At first, a

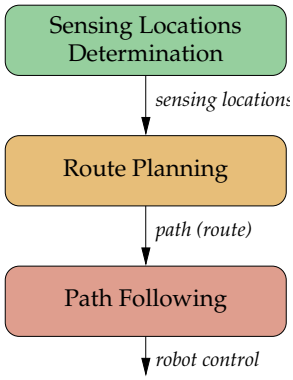


Figure 1.2: Decomposition of the inspection task.

set of sensing locations is determined as a solution of the AGP with respect to model of the robot and its sensing capabilities. The found set represents cities in the TSP, in which cities are connected by paths found by some path/motion planning technique with respect to the robot motion capabilities. A solution of the TSP is used to construct an inspection path, which is than followed by the navigation module of the robot.

1.2.1 A Group of Cooperating Mobile Robots

The above presented discussion of the inspection task considers only one mobile robot. The presented problems can be formulated for the multi-robot variants as the *Multiple Watchmen Route Problem* (MWRP) and the *Multiple Traveling Salesmen Problem* (MTSP). In both problems, two criterions can be considered:

- *MinSum* - minimization of the total length of the routes,
- *MinMax* - minimization of the longest route.

The MinMax variant is more appropriate for the inspection task in the context of search and rescue scenario, because the time to find possible victims is critical. If the total amount of spent energy have to be considered, the MinSum variant should also be taken into account.

The cooperative behaviour of mobile robots is considered by the MinMax criterion, which leads to assignment of the sensing locations to each robot. The coordination resolves problem of mutual robot avoidance and it can be efficiently solved if positions of the robots in time are known. The multi-robot planning with consideration of cooperation and coordination can be computationally unfeasible [176], therefore a decoupled planning can be applied. It means that collision free paths are found independently for each robot

according to optimization criterion. After that, velocity profiles for each robot are found to avoid possible collisions between robots. Alternatively if found paths do not intersect each other, the coordination is guaranteed instantly from the found paths.

1.2.2 Remark about Research Direction

The studied problem of the multi-goal path planning for a cooperating group of mobile robots in the context of the inspection task formulated as standard problems the AGP and the MTSP, or as the MWRP, is advantageous in several aspects. These problems are studied for a long time and several algorithms have been already proposed. For particular restricted problem variants, optimal solutions have been proved or class of approximation with the worst case or expected complexity have been proposed. The formal problem definition allows selection of an already available algorithm for the particular sub-problem.

On the other hand, the wide range of approaches is also challenging, because solutions for restricted class of sub-problems require careful consideration. The presented problem formulations are known to be NP-hard in general, therefore an extension of particular approach to solve more general problem can be too complex or not applicable at all. Moreover the robotic application of the inspection task have to be considered with high attention, because capabilities of real robots add particular constraints that are not considered by the computational geometry community (origin of the AGP and the WRP). Similarly the TSP is an optimization problem that uses a cost between cities. The cost is a path between two places in the environment, and the path have to be determined. Idealized algorithms for path planning must be augmented to deal with many annoying realities when applied in the field: motion constraints, complex definition of the goal, optimization criteria or side conditions and, as always, uncertainty [75]. Also from a practical point of view algorithms robustness should be considered. This is especially important for algorithms from the computational geometry, because even an optimal algorithm has been proposed and proved it can deal with robustness problems or the algorithm is not easy to implement [211].

To select appropriate approaches and to formulate and specify particular assumptions of the studied problem of multi-goal path planning in the context of the inspection task it is necessary to review not only the robotic domain, but also the computational geometry and the operation research domains. The real applicability of proposed solution of the cooperative inspection task strengthen the research direction to more practical approaches, thus it is not necessary to find the exact solution, which can be computationally expensive due to NP hardness of the problems.

1.3 Preview of the Thesis Contributions

The thesis follows the decoupled approach of the problem and provides particular contributions to decomposed sub-problems. In addition, the WRP and MWRP formulations are also considered. Particular contributions, and in same cases with already published papers, are as follows.

- Comparison of sensing locations algorithms and proposition of a new algorithm called *Boundary Placement*, Chapter 4. The preliminary results have been published in [85].
- Multi-goal path planning problem - TSP and MTSP-MinMax problem formulations:

  - comparison of soft-computing techniques [159, 156],
  - a new algorithm for the MTSP with a hierarchy of cities [88],
  - an extension of the self-organizing map (SOM) algorithm for the MTSP-MinMax to a polygonal domain, Chapter 5.
- Generalized multi-goal path planning problem for goals with a polygonal shape, Chapter 6:
  - a solution of the inspection task with segment sensing locations,
  - an application of the navigation functions in the SOM adaptation procedure,
  - an inspection planning for polygonal areas of interest.
- An algorithm for the WRP and the MWRP-MinMax with restricted visibility range in a polygonal domain, Chapter 7.
- Consideration of trajectory generation in the multi-goal path planning, Chapter 8:
  - consideration of acceleration limits [86],
  - comparison of RRT motion planners [264],
  - a new algorithm for the multi-goal motion planning problem called RRT–Path^{ext},
  - an algorithm for the MWRP with trajectory generation.
- An algorithm for the cooperative inspection task in a search and rescue mission [161].

1.4 Thesis outline

The structure of the thesis follows decomposition of the inspection task. The studied problem is related to several domains, which are studied for several decades thus only a brief description of the most related and inspiring approaches are presented in Chapter 2. The overview of related work leads to more specific problem formulation and thesis goals presented in Chapter 3. Chapter 4 is dedicated to solution of the AGP part of the inspection task, which is considered as the sensor placement problem. New proposed algorithm is compared with existing algorithms in several instances of the inspection task. In Chapter 5, the TSP and the MTSP are studied with focus to the Self-Organizing Map (SOM) approach to solve the MTSP-MinMax as the most promising soft-computing technique to solve various inspection task variants. The first part of the chapter is dedicated to the description of the selected SOM approach that is then extented to a polygonal domain. The proposed MTSP algorithm is applied for generalized goals in Chapter 6. New proposed algorithm for the inspection task with restricted visibility range formulated as the WRP/MWRP is presented in Chapter 7. Trajectory generation in the multi-goal path planning is discussed in Chapter 8, in which a new motion planning algorithm for the multi-goal motion planning is introduced. The algorithm is then used in the proposed WRP algorithm to deal with motion constraints and to plan inspection trajectories instead of paths. Finally conclusion and remarks about future work are presented in Chapter 9.

Chapter 2

Related Work

The studied multi-goal path planning is tightly related to path and motion planning approaches, because they provide paths between two goals. The multi-goal problem arises in a robotic task, which aims to fulfill desired mission objective by visits of (eventually given) goals. It is the reason why an overview of robotic tasks is also part of the literature review. The multi-goal path planning can be formulated as the TSP, therefore it is related to routing problems. In addition, the inspection task is related to the WRP and the AGP studied in the computational geometry. Therefore these domains are also included in the presented overview of related work. The review follows the organization into robotic tasks, visibility problems, and routing problems. The domains overlap in the mobile robotics, thus the division is not strict, especially for the computational geometry approaches applied in motion planning.

At this moment, it should be noted that path planning is sometimes called motion planning (and vice versa) in literature. These two problems are tightly connected, but a difference can also be found. The motion planning can be considered as a problem to find a trajectory, thus it also find a path. The reverse is not so straightforward, due to necessity to find a velocity profile along the found path, which can be non-trivial. Motion planning is therefore considered in the overview rather than "pure" path planning.

The chapter is organized as follows. Brief overview of environment representations is presented in the next section. Mobile robot navigation tasks are discussed together with review of the path and motion planning approaches in Section 2.2. An overview of the art gallery and watchmen route problems is presented in Section 2.3. Section 2.4 is dedicated to routing problems. A conclusion leading to research directions, problem specification, and thesis goals is presented in Section 2.5.

2.1 Environment Representation

A model of robot surrounding environment is called a *map* and it can be represented in various forms. The navigation task operates over the free space of the environment and representation of the occupied portion of the environment is necessary to recognize regions or locations in the environment. Methods to solve the path planning problem are designed to work efficiently with specific type of environment representation. In [75], three classes of environment representations are discussed: *spatial decomposition*, *geometric representation* and *topological representation*, which are briefly described in the next paragraphs.

Spatial Decomposition - The idea of the spatial decomposition is to represent space itself rather than to represent individual objects within it. The spatial decomposition is a direct sampling of the environment into two or more dimensions. Two dimensional grids are called the *occupancy grids*. The main advantage of this representation is its generality, the grids can represent anything. Grids are commonly used for data fusion in the mapping task, because they allow straightforward data fusion that deals with uncertainty e.g. the Bayesian approach [254], Dempster-Shafer Belief Accumulation [129], statistical decision theory [144] and histogram grids [34]. The only disadvantage is a high space complexity, therefore only limited regions can be represented by grids. An alternative representations based on cells are recursive hierarchical structures, e.g. *quadtree* or *octree* in the case of three dimensions. Hierarchical representations are suitable for environments where most of space is free or occupied, because the worst case is complete subdivision into smallest cells. A survey of hierarchical data structures can be found in [227]. From the path planning point of view the spatial representation is suitable for search-based techniques [272].

Geometric representation uses geometric primitives such as points, lines, curves and volumes. A two dimensional map of the closed environment is commonly represented as a polygonal domain where border polygon is used to bound the environment and obstacles are represented by holes. Polygonal representation is typically assumed in computational geometry, in approaches to solve the AGP, the WRP and related problems. From the mobile robotics point of view, grid based approaches are currently dominant in exploration and mapping tasks. Mainly because occupancy grid is able to deal with uncertainty, however it is computationally infeasible for large scale environments. Geometric models are use for path planning and for simple environments models can be robustly constructed from the grid by computer vision techniques [70]. A polygonal map can be created from a CAD model in an automatic or a semi-automatic way, because these models do not contain noisy data and processing is straightforward. Recent results in the geometric mapping are promising and provide polygonal maps from grids [23] or set of points [164, 170]. In [125], a vector representation of explored part of environment called *sketch map* has been used in the multi-robot mapping scenario. An overview of geometric mapping approaches can be found in [258].

Topological representation describes geometric relationships between the observed features rather than their absolute positions [112, 180]. The resulting representation takes a form of graph where nodes represent the observed features and edges represent the relationships between the features. The topological maps can be built and maintained without any estimates of robot position. It allows integration of large area maps without suffering from accumulated odometry errors, since all connections between nodes are relative, rather than absolute. A combination of topological representation with local metric maps is used to represent large scale outdoor environments [35]. The graph representation is very efficient for path planning approaches based on graph-search techniques.

2.2 Path Planning and Mobile Robot Navigation Tasks

The mobile robot navigation is part of all robotic tasks like exploration, inspection, coverage, or pursuit-evasion as it provides ability to move a robot towards a desired goal. The navigation consists of a set of techniques that provides localization of the robot, path planning, environment modeling and interpretation of the model. In addition, algorithms to control the robot movements are necessary to act in the environment. From this point

of view a mobile robot navigation is a complex problem, which is solved by several techniques from various domains. The essential capability of the mobile robot is its movement and to move a robot in a desired fashion the path planning problem is fundamental.

The general path planning problem is to find a collision free path from a given start position to a desired goal position in the environment. A path can be found for various criterions e.g. maximal clearance, allowable turning radius, travel cost. The most common approaches try to find the shortest path. Path planning algorithms are developed according to different theoretical assumptions and requirements concerning the following [75]:

- *Environment and robot* - the structure of the environment, the robot capabilities and shape.
- *Soundness* - is the planned trajectory guaranteed to be collision free?
- *Completeness* - do the algorithms guarantee to find a path, if one exist?
- *Optimality* - the cost of the actual path obtained versus the optimal path.
- *Space or time complexity* - the storage space or computer time taken to find a solution.

Path planning algorithms are used in particular navigation task, which can be divided according to a priori known information about the working environment. The exploration task is focused to create a map of the environment, while the inspection task deals with efficient planning of a path to "see" the environment. The coverage task can be viewed as a special case of the inspection task, where sensing range is limited to the size of robot, which is navigated to visit each point of the environment. However all these tasks can be constrained for static or dynamic obstacles, the moving objects are typical for pursuit-evasion problems.

2.2.1 Motion Planning

In every task mention in the previous paragraph some kind of path or motion planning is necessary as well as an algorithm to follow the found path. The studied multi-goal path planning problem for group of cooperating robots is more part of the Artificial Intelligence domain unlike the path following problem, which is more related to robot control. The general formulation of the motion planning problem is known as the *Piano Mover's Problem* and it is known to be PSPACE hard even in case of one robot [220]. The problem is to find a continuous path for a given starting and goal positions of rigid robot modeled as polyhedron inside the Euclidean space such that the path avoids polyhedral obstacles, or report that such path does not exist.

One of the most straightforward motion planning methods is consideration of the shortest path from one position to desired one. The method was used to navigate early mobile robot SHAKEY [201]. For a polygonal environment the shortest path is found in the visibility graph by Dijkstra's algorithm. For simplicity a point robot can be directly assumed, but it can be unrealistic consideration. To guarantee the path is obstacle free for a non-point robot, obstacles can be expanded by particular distance in certain direction. An approach based on expansion of obstacles by a radius of the disk robot has been proposed in [184] and it was one of the first steps to formulate the notion of the configuration space [176]. For simplicity assume two dimensional environment, a world $\mathcal{W} = \mathbb{R}^2$, obstacles $\mathcal{O} \subset \mathcal{W}$ and rigid body of robot $\mathcal{A} \subset \mathcal{W}$. The configuration space $\mathcal{C}$ is a set of all possible configuration of the robot $\mathcal{A}$. A configuration q, $q \in \mathcal{C}$ can be for example triple of parameters $q = (x, y, \varphi)$, where x and y denote position of the robot and φ its orientation. The obstacle region $\mathcal{C}_{obs}$ can be defined as $\mathcal{C}_{obs} = \{q | q \in \mathcal{C} \wedge \mathcal{A}(q) \cap \mathcal{O} \neq \emptyset\}$. The free space

is $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$. $\mathcal{C}_{free}$ is considered to be open, however to solve some optimization problems (like the shortest path) the closure $\mathrm{c}\,l(\mathcal{C}_{free})$ is used.

The shortest path approach can be called the *shortest-path roadmap* and it is one of the combinatorial approach in which a solution is found in a graph called *roadmap*. A roadmap can be found by cell decomposition, e.g. *trapezoidal decomposition* or *triangular decomposition*. The *Voronoi Diagram* can be used [31] to guarantee maximal clearance along a path. To deal with segments and not just points a generalization of standard Voronoi diagram (GVD) is more appropriate. The GVD can contain unnecessary information that is why hierarchical GVD has been introduced in [265]. A combination of the visibility graph and Voronoi Diagram called Visibility-Voronoi Complex [267] is suitable if an allowable clearance is given. The unifying concept of diagrams is called *Abstract Voronoi Diagrams* [152].

The above roadmap methods are suitable for a polygonal representation of environment, but they can also be applied to a grid. To find a path, e.g. the shortest one, any search based technique can be used, the most general are the $A^\star$ algorithm or Stentz's $D^\star$ variant. For a cell based representation the *Distance Transform* algorithm, introduced in 1966 by Rosenfeld and Pfaltz, can be used. It is propagation algorithm based on wavefronts increasing distance. In 1984, Jarvis adopted this algorithm for planning a collision-free path by starting propagation of the wavefront from the goal points to fill all of the free space, flowing around obstacles [133]. The main advantage of the algorithm is its simplicity and flexibility to deal with uncertainty or unknown prior information about the environment. The algorithm has been used for a variant of the pursuit-evasion task called *covert robotic navigation*, where it is combined with visibility computation in the so-called *Dark Path* algorithm [188]. Like other grid based technique, it is computational intensive, however this issue can be addressed by a hardware implementation on the FPGA [246].

The Distance Transform algorithm is kind of *wave-front* path planner [52]. Such a planner is also the numerical potential field technique [27]. A potential function is a differentiable real-valued function g from $\mathbb{R}^m \to \mathbb{R}$ and its gradient $\nabla g(q)$ points in the direction of increasing g. This function is used to provide a direction for robot movements and can be viewed as a vector field with attractive and repulsive forces. Desired goal attracts a robot while obstacles act as repulsive forces to avoid robot collision. The navigation of the robot is following negated gradient of the potential function $\dot{q}(t) = -\nabla g(q(t))$ and the robot stops when it reaches the goal $\nabla g(q) = 0$ [52]. If the function $\nabla g(q)$ has only one global extreme it provides suitable navigation function. However early approaches can trap a robot into local extreme, a harmonic potential function resolves this issue [58]. The harmonic function is a function that satisfies Laplace's equation $\nabla^2 g(q) = 0$. The solution of the equation can be found numerically by the finite difference method or the finite element method [57]. Combination of potential field functions with the roadmap based approaches can be found in literature. A combination of Voronoi diagram with locally used potential field method is proposed in [189].

Roadmap based approaches become impractical if dimension of $\mathcal{C}_{free}$ increases, mainly because they rely on explicit geometric representation of $\mathcal{C}_{free}$ [52]. Based on an observation that a test if q is $q \in \mathcal{C}_{free}$ is relatively fast randomized approaches were proposed. Two main approaches are *Probabilistic Roadmap Planner* (PRM) introduced in [146] and *Rapidly-Exploring Random Trees* (RRT) introduced in 1998 by LaValle. The PRM is two phase algorithm: the first phase is called a learning phase and the second a query phase. In the first phase, a roadmap is created from random samples. The samples from the free space, $q \in \mathcal{C}_{free}$, represent nodes of a graph. An edge between two nodes represents a path between nodes, the path is found by a local planner, e.g. straight lines that are entirely in

$\mathcal{C}_{free}$. The query phase finds a path for arbitrary start and goal configurations in the created graph from the first phase. The PRM based methods have been successfully applied for high-dimensional spaces providing high-quality paths [104]. A supporting corridor is utilized in the recent Corridor Map Method [105] to reduce required computational time.

The quality of path relates to its smoothness. A robot can follow only suitable path, which respects its kinematic and dynamic constraints. Such a path can be called a trajectory that can be directly used by robot motion controller. The term trajectory is denoted to a path parametrized by time and velocities, and accelerations can be computed from the first and the second derivatives with respect to time [52]. It should be noted that smooth shortest path can be composed from motion primitives, called Dubin's curves, even in case of obstacles [131].

Rapidly Exploring Random Trees

The RRT is a single-query planning algorithm that produces trajectories directly. It starts from an initial configuration $q_{init} \in \mathcal{C}_{free}$ and extents a tree of configurations towards the desired configuration q_{goal}. The tree represents configurations that are reachable from q_{init}. The RRT is probabilistically complete and it was developed for kinodynamic planning [178]. Besides, it has been successfully applied in other path planning problems [177]. Consideration of kinodynamic constraints (velocity and acceleration bounds) increases planning difficulty with dimension of $\mathcal{C}$ and problem constraints. The RRT provides a feasible trajectory, which is not typically optimal, but it can be locally improved.

The fast exploration of the RRT is based on the so-called *Voronoi bias*. If Voronoi diagram of the current expanded tree (diagram of the tree nodes) is created, then a new random sample will likely be in the largest region, hence the tree is more probably expanded in that direction. Many algorithm variants have been proposed during last years. They increase performance and address the main issue of the probabilistic planning methods: the *narrow passage problem*. Selected approaches are briefly described in the next paragraphs.

Exploration capability in the narrow passage problem is increased by several local trees in [245]. A new tree is started from the current random state if a tree vertex cannot be expanded towards the new state. The tree is expanded in parallel with the main tree and other trees to the goal. In [218], a combination of the RRT and cell-decomposition is proposed to addressed the problem.

The quality of a found path is discussed in [256]. A heuristic is proposed to select an expanding node that will provide path with lower costs. Instead of the closest node to the new random state, several nodes are considered (by the k-nearest neighbourhood function) and the best potential node is selected.

The sampling strategy based on visibility region of the tree node is used to dynamical adaptation of sampling domain in [280, 132]. A node with high number of fails of expansion attempts is considered to be close to obstacle and its sampling domain is decreased. Similar technique to local search tree is used in the RRT–blossom algorithm [142], which uses a local flood fill mechanism based on instantiation of all possible node expansions.

Ferguson and Stentz proposed re-usage of previously found tree to decrease required computational time and to increase quality of found paths in [92]. The node selection process is based on combination of distance to the goal and cost from the start to the node. Selected promising nodes from the previously found tree are considered in the new tree, while the cost from the start to the node is increased and weight of the distance to the goal is reduced by given values. The proposed technique leads to prefer costly solution

in the first tree to find a feasible path, which is then possibly refined to be cheaper. A RRT based algorithm is used for real-time path planning for the RoboCup soccer in [38]. The authors proposed a cache mechanism to store found paths that are used in the next planning iteration, because the environment is not significantly changed. The achieved movements was about 0.8 ms^{-1} and 1.7 ms^{-1} if the found path has been post-processed.

The required computational time of RRT algorithms is related to the computational efficiency of determination of the closest configuration from the tree to a newly sampled configuration. The speed of the search can be dramatically increased by an appropriate spatial structure, like the KD-tree for the Euclidean spaces. Extensions of the KD-tree structure suitable for topological spaces in the motion planning problems are proposed in [279]. The extension is available as an open-source library called the MPNN [278], which is based on the ANN library for the Euclidean spaces proposed by Aray and Mount [198].

The RRT based path planners have been successfully used in various robotic tasks. For example it has been used for path planning with uncertainty in the Particle RRT [191] or RRT-SLAM for the exploration task [128]. Successful motion planning for a car vehicle in DARPA Urban Challenge is reported in [162]. An application of the RRT based path planning for the multi-robot cooperative box-pushing is presented in [209].

For ability of the RRT algorithm to deal with various constraints and assumptions it is considered to be one of the most promising recent motion planning approaches.

Multi-Robot Planning

To plan a collision free path for several mobile robots two types of collisions have to be considered: robot–obstacle and robot–robot. A motion planner for a robot consisting of multiple bodies can be used to solve the multi-robot case [176], e.g. the RRT or the Sampling Based Roadmaps Trees (SRT) [52]. The dimension of the problem grows linearly with the number of robots, hence the complete algorithms require at least exponential time, see problem formulation in Section 7.2 [176]. Two approaches for the multiple robot problem are defined in [52]:

1. *centralized planning* - considers different robots as one multi-robot body,
2. *decoupled planning* - is two stage algorithm. At first collision free paths are found independently, after that the coordination is achieved by computing the relative velocities of robots along their individual paths.

An advantage of the decoupled method is that it does not increase dimensionality of the configuration space. On the other hand, it is incomplete, because even in the case both parts are complete, it may be impossible to find a collision free plan for two paths found in the first stage.

LaValle discussed other reason to separated study of multi-robot planning problem in [176]. If an optimality is important and performance of each robot is optimized, there is no clear way how to combine several objectives into a single optimization problem without losing some critical information. The Pareto optimality is proposed as the appropriate notion of optimality planning for the multi-robot motion planning.

2.2.2 Coverage, Inspection, Pursuit-Evasion and Related Tasks

The main problem motivation of this thesis is the cooperative inspection task in a search and rescue mission. Beside the motivation, the general multi-goal path planning for co-

operating robots is also related to other robotic tasks. An overview of several related approaches for robotic tasks is presented in this section, partially as overview of possible applicable approaches and partially as possible problems for further applications of eventually new proposed solutions.

Four main tasks are studied in the field of mobile robotics: exploration, coverage, inspection and pursuit-evasion. These tasks can be divided according to possible communication between robots into two types of planning categories: centralized and distributed [75]. Distributed systems are necessary for systems where communication is incomplete, which seems be natural choice for real robots systems. Multi-robot systems are described from distributed artificial intelligence point of view in [190] and a survey of distributed robotic systems with focus to basic robotic tasks are presented in [212, 18]. Both categories of planning are applicable in the motivation task, however the centralized approach is more appropriate, as the communication between members of the rescue team with the chief is mandatory, therefore the communication is not the main issue of the studied path planning problem.

The exploration task represents problems with a priori unknown environment. In this thesis, an environment is assumed to be known and even in the case an environment is changed during the search and rescue mission, it is still partially known. Newly gained knowledge about the environment in a form of map can be used for an update of prior knowledge and used for re-planning like in the case of known environment. The exploration task deals mainly with SLAM techniques, which are not part of this thesis, therefore these approaches are not explicitly studied. An overview of approaches can be found in [84]. The reset of this section is concerned to coverage, inspection and pursuit-evasion.

Coverage Tasks

One of the main ideas behind coverage path planning is an environment decomposition into a set of cells. Several decomposition methods have been proposed: the trapezoidal decomposition [173], the boustrophedon decomposition [51, 172], the minimal sum of altitudes decomposition [126]. Connections of the cells are represented by an adjacency graph, thus the planning task is formulated as the TSP. An on-line version of the coverage algorithm is proposed in [271], which uses on-line detection of landmarks. The working environment is decomposed into a collection of non-overlapping sub-regions. While covering a sub-region neighbouring sub-regions are detected. Detected sub-regions are subsequently covered to perform complete coverage.

A boundary coverage can be viewed as a special coverage problem [134]. It is also related to the exploration, where boundary (of obstacles) are used to determine known an unknown parts of the environment. The problem for multi-robot case is formulated as the k-Rural Postman Problem on a graph in [79]. Edges of the graph represent paths along which the robot has to travel in order to inspect a boundary segment, or paths needed to be traversed in order to move between disconnected inspection areas. The approach is extended in [269] by a re-planning technique to consider changes in the size of the robot team or the environment.

Inspection Tasks

The inspection task deals with searching an object (or more objects) in an environment. If positions of the objects are known, the task can be formulated as the multi-goal path planning problem. A PRM planner to find paths between goals is used in [225]. The main

idea of the approach is to decrease the number of required paths between goals in the TSP formulation of the problem. The proposed heuristic algorithm is based on computation of the minimum spanning tree (MST) to estimate which paths have to be computed. The algorithm outperforms the naïve algorithm presented in [244], however in the worst case it also computes the quadratic number of goal–goal paths. An improved variant of the algorithm has been used in the generalized multi-goal motion planning problem where goals are partitioned into groups [226].

The MST is also used in [186] to find 2-approx solution of the multi-depot variant of the MTSP. Motion constraints for a car that is able to move forwards and backwards are considered and results of Reeds and Sheep [219] are applied instead of Dubins's curves.

In more general variant of the inspection task, an Area of Interest (AoI) can be consider rather than a point goal. Genetic Algorithm methods to address problem of cooperative visits of Areas of Interest (AoIs) is used in [281] and a solution of the problem with four robots and 32 mines in the demining task is reported. The problem is formulated as the MTSP, where areas represent vertices and an edge represents cost according to distance between areas. The problem of visiting AoI is formulated as the task allocation problem in [163]. Authors showed that the problem is NP-hard and proposed heuristic algorithm called PRIM ALLOCATION and its modifications to deal with dynamic environment. Also the problem of path planning for an Unmanned Air Vehicle (UAV) leads to the TSP formulation if the AoI are isolated and purely visible [153].

Pursuit-Evasion Problems

The pursuit-evasion problem is similar to the inspection task in the sense of finding an evader in the environment by several pursers. In the inspection task, the object is assumed to be static, while the evader can move arbitrary fast and the speed of pursers is limited. If at least one purser see the evader, searching is finished. The pursuit-evasion (or the inspection task) can be called *searching in a polygon*, if the environment is represented as a polygonal map. According to analysis of the pursuit-evasion problem two approaches can be found in literature: the worst case analysis and probabilistic analysis [47].

Guibas at el. [113] discussed required number of pursuers $H(F)$, where F represents the closure of the collision free space, and pursuers have 360° unlimited vision. The problem of determining minimal $H(F)$ for a polygonal environment is NP-hard, and an upper bound of the minimal number of pursuers for a general space with n edges and h holes is $O(h + logn)$. An algorithm for one pursuer in a polygonal map and its extension for multiple pursuers are proposed in [175]. The algorithm is based on decomposition of a polygon into conservative cells.

An operator $AH(D)$ *angle hull* for analyzing an on-line exploration strategy of an unknown environment is developed in [121]. The angle hull of a region D is a set of all points in the polygon P that can see two points of D at the right angle. The operator is used in an algorithm for a simple unknown polygon P. The algorithm finds a tour that is less than 26.5 times long than the shortest watchman tour computed off-line.

A probabilistic approach combining the pursuit-evasion problem and the exploration task is presented in [120]. Authors considered one or several evaders, limited precision of sensors and a prior probabilistic map, which can be made from inaccurate information. Two greedy pursuit policies called: local-max and global-max are proposed in [259] to maximize probability to capture the evader. An environment is represented as occupancy grid and the local policy considers only reachable cells, while the global policy considers

the whole map. An approach to minimize needed time to find an object in the known environment is proposed in [228]. The position of the object is unknown, but it is characterized by the known probabilistic density function. A continuous sensing approach is used to determine sensing locations and a sequence of visits. The approach has been extended for several robots in [229].

The pursuit-evasion problem on a graph representation is studied in [130]. The evader visibility is limited and can see only adjacent nodes. A polynomial algorithm for two pursers and one evader is provided. A graph is also used in the multi-agent approach to the path planning problem in [155]. Nodes of the graph represent certain places of the environment and each node has associated information about movements in the time. Edges represent admissible paths and their weights are changed to avoid collision. Several expert rules are used to guarantee collision-free path planning.

A distributed algorithm to deploy agents with line-of-sight sensing is proposed in [102]. The algorithm is based on a graph called vertex-induced tree. An environment is represented as a simple non-convex polygon, which has to be fully guarded. Nodes of the graph are star-shaped polygons created as visibility polygon of the polygon vertex. Two nodes are connected by an edge if their star-shaped polygons share an edge. The deployment algorithm is local navigation algorithm, which runs asynchronously on each agent.

2.3 Art Gallery and Watchmen Routes Problems

The Art Gallery Problem (AGP) as well as the Watchmen Route Problem (WRP) belong to the class of visibility (also called illumination) problems. The class contains several interesting problems and proposed solutions related to robotic navigation tasks, especially if a polygonal map is used to describe an environment. That is why a survey of visibility problems is presented in this section.

The AGP was originally posed by Victor Klee in 1973 and the most basic form is [122]: "*What is the smallest number of guards needed to guard an art gallery?*". The problem is defined on a polygonal representation of the environment and notion of visibility is necessary to problem definition. Two points in a polygon P are called *visible* if the line segment joining them is contained in P [257]. Formulation of the AGP for a polygon P is to find a minimum set of points G (guards) in P such that every point of P is visible from some point of G. A sufficient number of guards is $\lfloor \frac{n}{3} \rfloor$ for a simple polygon, which has been proved by Chvátal [53]. The proof is called the *Art Gallery Theorem* and it has been simplified by Fisk in 1978 [93]. The idea of his proof is based on the triangulation of polygon and 3-coloring of polygon vertices, such that each triangle has all three colors. The polygon is guarded if all triangles are guarded, because a triangle can be guarded by one of its vertex, a color with the fewest vertices denote the guards.

Variations of the AGP where a guard is a subset of the polygon instead of a point have been proposed. Typical types of the subsets are polygon edges (*edge guards*) or convex sets. In 1981, Toussaint started investigation of the AGP with mobile guards. The guards were allowed to patrol individual edges of a polygon rather than standing at the same point. O'Rouke proved that the minimal number of mobile guards necessary to guard any polygon of n vertices is $\lfloor \frac{n}{4} \rfloor$ [204]. In [235], Shermer investigated diagonal guards, which are allowed to patrol a segment between nonadjacent vertices of polygon. The number of required guards has been bound for several classes of polygons. Orthogonal polygons can always be guarded by $\lceil \frac{n}{4} \rceil$ guards and $\lceil \frac{n}{4} \rceil$ guards are sometimes necessary. Czyzowicz

et al. proposed that a rectangular art gallery with r rooms can be guarded by exactly $\lceil \frac{r}{2} \rceil$ guards [67]. A rectangular gallery can be divided into rectangular rooms. If the rooms share an edge, which represents a door between rooms, the guard can then stand in the door between the rooms.

A polygon P with n vertices and h holes can always be guarded with $\lceil \frac{n+h}{3} \rceil$ point guards [32]. A guard is called vertex guard if its position is restricted to vertices of P. A polygon with holes can be always guarded by $\lfloor \frac{n+2h}{3} \rfloor$ vertex guards [205]. Shermer proved that for h=1 the bound is $\lfloor \frac{n+h}{3} \rfloor$. An exact polynomial algorithm for orthogonal polygons has been proposed in [61]. It is based on discretization of simple polygon P and solution of instances of discrete Set Cover problems, which is formulated as the Integer Programming problem. This approach has been also applied for approximate solution of the AGP for simple polygons in [62].

Liaw, Huang and Lee proposed notion of *co-operative guards*. A set of guards is called co-operative if they guard whole polygon P and their visibility graphs are connected. The problem is NP-hard for a simple polygon. A sufficient number of co-operative guards for a simple polygon with n vertices is $\lfloor \frac{3n-1}{7} \rfloor$ [192]. The *k-guarded guard* is a guard that is visible from other k guards. An orthogonal polygon with n sides can be guarded by $k\lfloor \frac{n}{6} \rfloor + \lfloor \frac{n+2}{6} \rfloor$ k-guarded guards for $k \geq 1$ and $n \geq 6$.

The problem of finding a minimum number of guards is NP-hard for a polygon with holes. Furthermore, for point, vertex and edges guards the solution cannot be approximated by any polynomial time algorithm with a ratio of $\frac{1-\epsilon}{28} logn$ for any $\epsilon > 0$ unless NP $\subseteq$ DTIME($n^{O(loglogn)}$), where DTIME(t) is the class of all sets accepted by deterministic Turing machines whose running time is bounded by $t(n)$ [81].

The guarding problem is related to covering problems. Guarding a polygon by guards chosen from some set S can be viewed as covering the polygon by visibility polygons of members of S. Two problems are studied: the *convex partitioning* and the *convex covering*. The difference is that the partition deals with convex set of polygons with disjoint interiors, while polygons can have joint interiors in the covering problem. Covering of an orthogonal polygon by minimal number of rectangular polygons is NP complete even for a polygon without holes [65]. The authors of [197] proposed a polynomial algorithm with $O(n^{10})$ for covering orthogonal polygons by star polygons. The convex partitioning is NP-hard for polygons with holes [183].

Laurentini considered the edge-covering problem in [174]. The problem is similar to the AGP, but instead of covering polygonal areas only edges (boundary of objects) have to be covered. This problem is an important research area in computer vision, it is part of the inspection problem or reconstruction of the object model. In [36], authors proposed a restriction that each edge of the polygonal object must be observed entirely by at least one guard. The restriction allows to find one or more sets of regions where a minimal set of view points can be independently located. Upon this restriction the authors proposed algorithm that asymptotically converges to the optimal solution of the unrestricted problem. The algorithm is extended for 3D environments in [37].

A set of heuristics were proposed and tested in [13] for the AGP with unrestricted visibility capabilities of guards. At first, a candidate set of guards is constructed, then heuristic selection is applied to find smaller guarding set. An approximate solution of visibility-independent set problem is proposed. It provides a lower bound of the required number of guards. A visibility-independent set is a finite set, $I \subset P$, of points in the polygon P such that visibility polygons of points $p \in I$ are pairwise disjoint.

A variant of stationary guards searching for a mobile intruder, called *k-searchers*, has been introduced in [248]. These searchers can see in k directions. The ∞–*searchers* have 360° field of view. An example of 1–searcher can be a guard with a flashlight. Used visibility allows "line of sight" to "gaze" a reflex vertex (or an edge) of δP as long as it does not go outside P, where P is a polygonal region that includes its boundary δP [276]. An $O(n^2)$ algorithm to decide whether a simple polygon with n edges can be guarded by an 1–searcher was introduced in [179]. If a polygon can be guarded, the algorithm provides a search schedule. The 1–searchers and ∞–searchers problem variants for a polygon with holes are NP-hard [275].

More realistic sensing capabilities of guards can be considered by additional visibility constraints. Two algorithms determining minimal number of sensing locations with consideration of restricted visibility range and incident angle are proposed in [108]. The restricted visibility range to a distance d can be stated as follows: two points p and q in a polygon P are called *d-visible*, if the line segment joining them is contained in P and the segment length is less or equal to d. The incident angle constraint regards a situation where a guard prefers to watch a scene directly rather than under unsuitable angle [250]. The problem is called *sensor placement* rather than the AGP to emphasize the visibility constraints. Proposed approaches are based on sampling of the environment [108]. The first algorithm uses a greedy algorithm to find a near-optimal subset of sampled points to cover boundary decomposition with complexity $O(nm^2)$, where n is the number of edges and m is the number of samples. The second algorithm firstly places a point p at the boundary of the unseen part of the polygon and then selects a point with the highest coverage from sampled points in the visibility polygon of p. The complexity of the algorithm is $O(mn_gnlog(n_gn))$, where m is the number of samples in a visibility polygon, n_g is the number of found guards and n is the number of edges. The algorithm was applied to determine a set of guards in a simple polygon for the inspection task in [69]. The inspection path was found as a solution of the TSP to visit the found set of guards, the proposed approach has been also extented to 3D representation of environment.

Sensing with 360° field of view of the panoramatic camera was described in [147]. A sufficient detail of objects on a captured image limits the camera visibility range. The proposed method decomposes a simple non-convex polygon with holes into to a set of convex polygons. A convex polygon from the collection of convex polygons is then divided into convex sub-polygons such that each sub-polygon can be guarded by one guard with limited visibility range. The algorithm can be used for an inspection of workspace borders and for the complete inspection of the environment. The issue of the proposed algorithm is that some unnecessary guards are placed if neighbouring convex sub-polygons have size close to the visibility range. However the found solution is only sub-optimal, the algorithm is very fast. The required computational time increases proportionality to the number of found guards and authors reported problem solved up to 8000 guards for restricted visibility range.

Real surveillance multi-camera vision systems require additional visibility constraints. In these systems, cameras are typically mounted on walls and have limited field of view (*viewing frustrum*) and limited range for sufficient sharp focus in an image. The problem formulation considering these limitations is proposed in [82] and a solution based on discretization of environment into two dimensional grid is presented in [83]. The Binary Integer Programing (BIP) formulation of the problem with similar constraints is proposed in [124]. Four problems are studied, maximization of coverage for given number of cameras of same type, maximization of coverage for several types of cameras and given

total cost, maximization of coverage for given number of cameras and their possible positions, and minimization of the cost for a given required coverage. Their exact solution is compared with a greedy heuristic and the randomized dual sampling algorithm [108]. Authors reported suitability of heuristic approaches and its well approximation of BIP solution, however due to the lack of computational resources only small problems have been evaluated. Similar problem has been addressed in [207]. The problem is the optimal sensor placement to create wireless sensor network where each sensor has three parameters: sensing range, field of view, and orientation. A limited bandwidth of a wireless link is also considered. The problem is formulated as integer linear programming model and solved by CPLEX 10.1 solver running on 3 GHz CPU. The largest solved problem has 200 placement sites and 70 control points and was solved in 30 826 seconds.

The graph theory was applied to describe properties and relations of visibility in polygons. The concept of the *visibility graphs* was introduced by Avis and ElGind in 1983. Two types of the visibility problems can be defined. The first type is the *pure visibility problem* that can be expressed by the formalism of the graph theory on *point visibility graph* introduced by Shermer. The second type is the *hybrid visibility problem* in which other information such as geometric or conceptual properties are necessary.

Watchman Route Problems

The *Watchman Route Problem* (WRP) is an example of the hybrid visibility problem with metric information. Chin and Ntafos defined the watchman route for a polygon P as a closed walk in P such that every point of P is visible from some point of the walk [49]. They proved that the problem is NP-hard for polygons with holes and proposed an $O(n)$ algorithm for an orthogonal polygon.

If a visibility range is restricted to a distance d, two variants of the WRP can be found in literature [251]. The *d-watchman route problem* is a variant to see only the boundary of the polygon, while the *d-sweeper route problem* aims to sweep a polygonal floor using a circular broom of radius d, so that the total travel of the broom is minimized [202].

A variant of the WRP called *minimum-link watchman route* is studied in [22]. The criterion of the problem is to minimize the number of links of the route instead of the Euclidean distance. Authors considered a polygon with convex holes and proved that the problem is NP-hard and proposed a polynomial approximation algorithm.

Another variant of the WRP, motivated by the restricted visibility range, deals with a problem to find a route inside a polygon P that visits a given collection of subsets of P. If the collection consists of sub-polygons, two problems are studied: the *safari route problem* [202], and the *zoo-keeper route problem* [50]. Both problems are NP-hard in general. The collection is a disjoint set of convex polygons inside the polygon P in both problems. Each sub-polygon shares an edge with the polygon P. The route should not enter the interior of any sub-polygon in the zoo-keeper route problem. In the safari route problem, an enter of the route to the sub-polygons is allowed. Chin and Ntafos proposed an $O(n^2)$ algorithm for the zoo-keeper route problem for a simple polygon. An $O(nlogn)$ algorithm based on the full shortest path map is proposed for the zoo-keeper route problem with given starting point in [30]. A polynomial algorithm $O(n^3)$ for the safari route problem in simple polygon is presented by Tan and Hirata in [251].

The *touring sequence of polygons* is studied in [74]. The problem is to find a route that visit sequence of polygons. Two bounds are proposed: an $O(n^2logn)$ bound for the safari route problem and an $O(n^3logn)$ bound for the WRP with a fixed starting point and a

simple polygon. The *aquarium keeper problem* deals with a problem to find the shortest closed path inside a polygon that visit each edge at least once. Czyzowicz et al. proposed a linear time algorithm for the aquarium keeper problem for a simple polygon in [66].

If a guard moving along the shortest watchman route surveys a polygon only at some selected points, the problem is close to the decoupled approach of the inspection task with discrete sensing. Authors of [46] call the points as the *vision points* and proved that the problem of determining the minimum number of the vision points along the shortest watchman route is NP-hard. The problem for a street polygon is studied in [45].

The WRP in a polygon P is a special case of the *m-Watchmen Routes Problem* (MWRP) that aims to find a route for each of m watchmen such that each point of the polygon P is visible from at least one route. For m =1 the problem is the WRP. If m is so large that the total length of the routes is zero, the problem is the stationary AGP. Two variants of the MWRP according to minimized criterion can be found in literature: the MinSum variant minimizing the total length of the routes, and the MinMax variant minimizing the length of the longest route. Nilsson proved that MWRP is NP-hard even in simple polygons for both criterions [200]. He also investigated a problem for restricted polygons and proved polynomial algorithms for spiral polygons and histogram polygons.

Probably the first heuristic approach for the MWRP has been proposed by Packer in [211]. The approach is based on a set of static guards S found by heuristic A_1 of [13] and constructing minimum spanning tree of S. Distances between two guards are found as length of the shortest path from the visibility graph. The tree is split to m sub-trees (for m-watchmen) and Hamiltonian routes on each sub-tree are independently constructed. Vertices along route are substituted by others that shorten the length of the route and maintain full coverage. Finally redundant vertices of the route are removed. Both variants (MinSum and MinMax) are addressed by the proposed heuristics.

The WRP can be considered as a variant of the view planning problem (VPP) studied in robotics, which can also be considered as a variant of the inspection task. The VPP aims to provide a plan for an automated 3D object recognition and inspection. To provide a "good" view it is necessary to consider sensing constraints, which depends on particular sensing device and recognition capabilities. Regarding these constraints the VPP is more complex than the WRP [232]. Two costs can be consider in the VPP, the sensing cost and the travel cost from one view point to the next view point. The problem decomposition into the AGP and the TSP is criticized in [266]. Even in the case that both sub problems are solved optimally, because they are solved independently, the overall performance can be poor. Author noted that such decomposition works well if coverage of considered views do not overlap or those with large coverage overlap are closed to each other. The combination of view and travel costs is addressed in the formulated Traveling VPP, which is defined on given set of viewpoints. The solution (probably the first unified approach in robotics) is based on the ILP formulation and rounding algorithm called Round and Connect. The WRP is reduced to the Traveling VPP by a finite number of viewpoints that are found by the proposed sampling algorithm. The number of viewpoints does not depend on geometric parameters of the polygon (with holes), the viewpoints are found in $O(n^{12})$, where n is the number of polygon vertices.

Two extensive surveys should be explicitly mentioned [236] and [257] to conclude the visibility problems overview.

2.4 Routing Problems

If paths between goals are determined, or more precisely if costs between goals are defined, the multi-goal path planning problem can be formulated as a routing problem. The most famous routing problems are the TSP and its generalized version *Vehicle Routing Problem* (VRP), in which capacities of vehicles (salesmen) are restricted. An overview of approaches for the TSP and its multi-robot variant the MTSP is presented in this section. Besides, variants of problems that can be applied in the multi-goal planning are also included in the overview. A goal is called city in the TSP formulation, thus the term city is also used for goals in this thesis.

The Euclidean TSP, or the geometric TSP, is a variant of the TSP, in which a distance between cities is the ordinary Euclidean distance. In combinatorial approaches, the TSP is typically formulated on a graph $G(V, E)$ where V is the set of cities and E is the set of connections between the cities. Each edge has associated cost derived from a path (e.g. the shortest one) between two cities, or another metrics can be used, e.g. time to travel.

The TSP is known to be NP-hard and it is studied for a long time by the operational research community. Solutions of the TSP can be divided into two types: exact and heuristic. While exact algorithms require lot of computational power, due to complexity of the problem, heuristic algorithms provides approximate solution in a reasonable time. Two basic types of heuristics can be defined: constructive and improvement. The constructive heuristic starts with one city and gradually creates a route. The improvement heuristic starts with some route that is improved by local search techniques.

Comparisons of heuristic algorithms are presented in [138, 139, 140]. In these comparisons, the solution quality and effectiveness of various approaches are discussed. An influence of supporting data structures are discussed in [99, 208], and with respect to large problems in [17]. Successful approaches are based on λ–opt heuristics published in 1973 by Lin and Kernighan [182]. Their most powerful heuristic called *Lin-Kernighan* proposed in the same year has relatively recent efficient implementation made by Keld Helsgaun in 2000. Helsgaun discussed the effective implementation in [118] and described his implementation called LKH. However the heuristic algorithm finds an approximation solution, the optimal solutions are produced with high frequency. The largest problem solved by the LKH is the *World TSP* with 1 904 711 cities and the best reported length is 7 515 877 991 (May 12, 2009). The current best lower bound of this problem established by the Concorde TSP code with the CPLEX solver is 7 512 218 268 (June 5, 2007) [16].

Beside classical heuristics, modern heuristics like tabu search, adaptive memory [169] or variable neighbourhood search [196] have been proposed to improved quality of solution. The current research is focused on developing meta heuristics over the heuristics [33]. Meta-heuristics guide the search process, and they are non-deterministic and sometimes use search experience stored in the memory. These approaches include soft-computing techniques like: Ant Colony Optimization, Genetic Algorithms, Simulated Annealing and Neural-Networks.

The Ant Colony Optimization technique is based on distributed autocatalytic process that simulates behaviour of real ants [73]. Artificial ants use pheromone trails to share information among regarding paths Moving ant marks path by a trail of this substance. Other ants detect the pheromone on the ground and decide to follow path in varying quantities of the substance and reinforce the trail with their own pheromone. If ants are moving along the shortest path repeatedly from the start to the end the amount of

pheromone is stronger than on other paths, because ants frequency on trail reinforcement is higher.

Genetic Algorithms are stochastic algorithms that operate over population of possible solutions [193]. An individual of population is evaluated by a fitness function. The convergence of the best individuals are satisfied by a selection mechanism, crossover operators and mutation operators.

The Simulated Annealing exploits an analogy between the way in which a metal cools and freezes into a minimum energy crystalline structure (the annealing process) and the search for a minimum in a more general system. The idea is motivated by finding an equilibrium configuration of a collection of atoms at given temperature. The connection between combinatorial optimization and statistical mechanics was proposed in [151]. The algorithm consists of generator of random changes in solutions, evaluating the problem functions and an annealing schedule that is an initial temperature and rules for lowering it as the search progresses.

An application of the Neural-Network (NN) to the TSP was started by Hopfield's work in 1985 [123]. Followed by different approach called *elastic net* [77], which has direct geometrical representation as "rubber band". It is non-uniformly elongated by an iterative process until it passes all cities in sufficient distance. Smith [240] noted that before this work has been published, Fort [96] had been working on using Kohonen self-organizing process [154] to solve the TSP. In 1988, Angéniol et al. introduced an algorithm based on the self-organizing principles and reported solution of the Euclidean TSP with 1000 cities. Approaches based on the Kohonen competition and adaptation rules are called Self-Organizing Map (SOM) and have been used for various routing problems. The SOM procedure provides transformation from high dimensional data space to low dimensional map space, it principally deals with vectorized data, however generalization to function space has been introduced in [101].

The TSP as a benchmark for the Hopfield based neural-network is criticized in [238]. Mainly because the NN uses quadratic formulation, which consists of many local minima, while heuristic functions are based on the linear formulation, which has more constraints, but only one local minima, thus direct comparison is not fair. Author's critic was not advocacy of inferior solutions, it was observation that performance of the neural networks for solving practical optimization problems had been relatively untested [240].

The SOM approaches have been mostly applied to the Euclidean TSP, however several approaches address other routing and optimization problems [231, 239], or are based on a graph input [274]. Despite the fact that the performance of SOM approaches is worse in comparison to classical heuristics from the operational research domain, they show interesting results [56] and can be used to construct a tour that can be improved [43] using λ-opt procedures [182]. Theoretical properties of SOM have not been proved for general case [60], although the SOM approach has been widely used in various domains [216, 241].

The classical and modern heuristics are combined in distributed computational environment. Solutions can be found by distributed computation of the linear programming problem formulation or by multi-agent techniques. In [252], different agents are used for construction, improvements and refining of the solution. An application of the Multi-Agent Optimization System to the TSP has been presented in [273]. Each agent has only partial knowledge about the problem and agents are able to support a cooperative search.

One of the routing problem variant is a generalization of the Euclidean TSP called the *Traveling Salesman Problem with Neighborhoods* (TSPN). In this problem, each buyer spec-

ifies a meeting region. The buyer is willing to meet with a salesman within the region and the salesman wants meet all neighbourhoods. The TSPN is APX-hard and cannot be approximated within a factor $2 - \epsilon$, where $\epsilon > 0$, unless P=NP [224]. An approximation algorithm for the TSPN is presented in [76].

A special case of the TSP where a cost of edge depends on edges traversed earlier is called the *Angle Traveling Salesman Problem* (Angle-TSP). The problem has been introduced in Fekete's Ph.D. thesis [90] and it aims to find a tour of the points with a minimum total angle. Authors [12] showed that the problem is NP-hard. A problem to decide whether a set of n points in the Euclidean plane can be connected into closed directed tour consisting of straight line segments such that all angles between consecutive line segments are from a set $\boldsymbol{A} \subseteq \langle -\pi, \pi \rangle$ is called *Angle-Restricted Tour* (ART). A tour with angles from $A = \{\alpha | 0 \leq \alpha \leq \pi\}$ is called *pseudoconvex*. If $A = \left\{-\frac{\pi}{2}, \frac{\pi}{2}, \pi\right\}$ a tour is called *orthogonal*. A problem of pseudoconvex and orthogonal tours is discussed in [91]. The authors proposed a classification of the problem where tour is a part of orthogonal grid and showed that the ART problem for an orthogonal tour is NP-hard. Problems of finding a tour with the minimum length, the minimum number of turns or combination are studied in [21], in which orthogonal milling problem is shown to be NP-hard and several constant approximation algorithms are presented.

A problem to find a tour that visits each edge of the given graph is called the *Chinese Postman Problem*. The problem can be transformed into a TSP instance or solved directly. A direct approach for the problem with turn penalties is presented in [55]. The graph is assumed to be rectilinear, i.e. all edges are meet at 90° or 180° angles. In such case penalties can be zero for a straight crossing, two for a right turn, five for a left turn, and nine for U-turn. This assessment corresponds to complexity off crossroads passing by a car. The authors compared six strategies how to handle turn penalties.

The *Traveling Salesman Problem with Backhauls* is studied in [106]. In this TSP variant, a product is picked up and transported to the depot from some customers at a return path. Two sets of cities are considered: cities that have to be visited at forward part of the path and cities that have to be visited at return path to the depot. The algorithm is based on SOM, where a salesman route is divided into two chains that are adapted to the particular set of cities. Experimental results are reported for problem size up to 1000 customers and compared with the exact branch-and-bound method, GENIUS, and its modification based on the variable neighbourhood search [196]. The SOM is used as a constructive heuristic to find a tour which is then improved by the 2-opt procedure. Authors reported that the SOM approach is competitive with the branch-and-bound exact method and the best cutting heuristic from the solution quality point of view, but requires more computational effort. They also noted an advantage of SOM flexibility to address the TSP with Backhauls and remark limitation of the SOM for its inability to address non-Euclidean instances.

Multiple Traveling Salesmen Problem

An extension of the TSP for several entities is the *Multiple Traveling Salesmen Problem* (MTSP) that can be formulated as follows: find shortest paths for m salesmen starting from a given city, passing trough all cities and returning to the starting city. The starting city is called *depot* and other cities have to be visited only by one salesmen. The problem is also called the *m-Traveling Salesmen Problem* and it is known to be NP-hard. Similarly to the MWRP two criterions can be considered: the MinSum and the MinMax. A transformation of the MTSP to the TSP has been proposed in [29], but the solution of the transformed problem is highly degenerated for the MinMax variant of the primal MTSP [98].

The first attempt to solve the MTSP-MinMax exactly has been proposed in [98]. The algorithm is based on a solution of the *Distance Constrained* VRP. A solution of the MTSP is used as the distance constraint, which is gradually decreased, and if the VRP does not have a solution, the previous solution of the MTSP is the exact. A solution of the MTSP is found by the heuristic algorithm based on the tabu search and neighbourhood heuristics called GENIUS. The MTSP-MinMax for two salesmen without the depot is studied in [41]. Author proposed characteristic Boolean function for this class of problems that is monotonic and self-dual for complete graphs with metric distances.

The MTSP-MinMax on a tree is studied by Averbakh and Berman. The location-allocation variant of this problem deals with finding routes and also depots for each salesmen according to MinMax criterion. The problem is NP-hard for more than one salesman [24]. The exact solution of the location variant, the problem to find optimal depots, can be found in linear time for two and three salesmen [25].

The MTSP is an instance of the VRP with unlimited vehicles' capacities. There exist two main directions of heuristic approach to solve the VRP: constructive heuristic and 2-phase algorithms. One of the first constructive heuristic is the *savings* algorithm introduced by Clark and Wright in [54]. The problem is decomposed into clustering of vertices into feasible routes and actual route construction in a 2-phase algorithm. Both orders of steps are used : *Cluster First, Route Second* and *Route First, Cluster Second*. Three heuristics were proposed in [115], they are modified savings, modified sweep and Route First, Cluster Second. Heuristics are based on combination of GENIUS with the PRIMAL1 heuristic for the set covering problem and modified 2-opt procedure for the multi-vehicle problem. A tabu search heuristics are used to escape from a local optimum [169]. A hybrid approach based on combinatorial local search and evolutionary algorithms are presented in [213], author reported experimental results for a problem with 120 paper subscribers, one depot, and four newspaper distributors, a solution was found in thousands of seconds.

The exact algorithms can be divided into three main categories: direct tree search method, dynamic programming and integer linear programming (IP) [166]. A branch-and-bound algorithm and algorithm based on cutting planes are presented in [167]. In [168], authors reported experimental results for asymmetrical distance-constrained VRP up to 100 cities, the reported solution was found in hundreds of seconds. The modified branch-and-bound algorithm based on the *Concorde* TSP solver extented by the cutting planes technique to address the MinMax objective [15] was used to solve the Whizzkids'96 problem exactly. The solution was found in ten days running on 188 processors of distributed computing environment. Authors reported sum of processing times scaled to the 500 MHz Alpha EV6 processor, the computational time was about 72 millions seconds.

A logical framework for vehicle routing planning is proposed in [234]. Authors used temporal logic to formulate planning task for a single vehicle.

The SOM approach has been used for the Euclidean MTSP-MinMax in [243]. In [159] authors compared the SOM approach with other soft-computing techniques ACO and the GA in several non-Euclidean instances of the MTSP. Despite the used naïve approach of the SOM to deal with non-Euclidean distances, authors reported competitiveness of the SOM approach from the solution quality point of view and also from the required computational time point of view.

A comprehensive bibliography of the routing problems with five hundreds references can be found in [165] and an overview of the MTSP formulations and approaches is presented in [28].

2.5 Discussion and Research Directions

The presented overview shows relation of the studied problem with several research topics. The computational geometry is focused on classification of problems and particular exact algorithms for restricted class of problems. On the other hand, problems motivated by a robotic application typically assumed general representation of the environment and more realistic capabilities of the robot, which leads to additional constraints. It is clear that it is necessary to restricted studied problem by properly selected assumptions. The following selection of assumptions is based on the problem motivation and possible practical applicability of the proposed solution in the real rescue mission experiments, where real-time requirements have to be satisfied. In addition, the proposed solution should be flexible enough to address particular modifications of the problem related to the cooperation of humans and robots during a search and rescue mission.

The first assumption that have to be made is a suitable environment representation. The visibility problems are mostly studied for a polygonal domain, whereas occupancy grids are suitable for map building. Recent results in creation of a geometrical representation of the environment are very promising, and also the environment is known in the inspection task, therefore polygonal maps can be assumed.

Regarding an ability of approaches to deal with various constraints the flexibility of randomized approaches is a great advantage. The randomized motion planners seem to be very promising, the RRT is one of approaches that is able to address kinodynamic constraints. For the AGP or the sensor placement problem, the randomized algorithm [108] provides solution for a realistic physical constraints [109, chap. 48]. The restricted visibility range is a practical assumption for the related search and rescue mission as smoky environment can be expected.

The multi-goal path planning problem formulated as the TSP is a combinatorial problem where a path between cities represents a travel cost without direct relation to the environment and robot capabilities. To guarantee traversability of a path between cities it is necessary to solve the path planning problem between the cities. The TSP is NP-hard, and the path planning problem considered as the motion planning can be in PSPACE, therefore to make studied approach feasible and applicable it is necessary to consider a model of the robot simple enough, but still sufficiently close to real mobile robotic platforms. A robot with differential nonholonomic drive is such suitable model, because it represents commonly used research-platform for indoor experiments.

The problem of cooperative behaviour of several robots can be considered as an optimization problem how to allocate goals to particular robots to fulfill the inspection task. The MTSP formulation with the MinMax criterion seems to be a natural choice.

The SOM approach to solve the TSP, particularly MTSP-MinMax [243], has been considered interesting, because a solution is represented by a ring of nodes that evolves in the input space. A direct geometrical representation of the ring can be advantageous in the studied hybrid visibility problems where geometric properties are necessary. The idea of the ring evolving in a polygonal domain is the main reason why the SOM approach is selected as the main technique in the studied multi-goal path planning problem in this thesis, therefore a more deep review of SOM approaches for the TSP is presented in Chapter 5.

Chapter 3

Problem Specification and Thesis Goals

3.1 Problem Specification

Multi-Goal Path Planning in the Inspection Task:

Given a polygonal map of the environment find a set of paths for given number of mobile robots such that the whole environment (free space) is inspected (seen) by at least one mobile robot while robots are moved along the planned paths, the length of the longest path should be minimized, the visibility range of the robot is restricted and differential nonholonomic drive of the robot is assumed.

3.2 Thesis Goals

1. Propose a method suitable for the cooperative inspection task with discrete sensing based on the decoupled approach: the AGP and the TSP/MTSP-MinMax. Consider applicability of the method in a search and rescue scenario, where heterogenous group of mobile robots (entities) is mandatory. Each entity can have different moving capabilities. In addition, the method should address the re-planing problem, in which entities have different initial positions.
2. Consider applicability of the SOM approach for the MTSP-MinMax in additional cooperative tasks, e.g. problem of cooperative visits of areas of interest.
3. Consider applicability of the SOM approach to address the inspection task with continuous sensing, the MWRP.
4. Consider motion planning constraints in the studied multi-goal path planning problem.

Chapter 4

Finding Sensing Locations

Sensing locations are places in a workspace where measurements are performed. For the discrete sensing the number of sensing locations is finite and the problem is to find a set of sensing locations can be formulated as the Art Gallery Problem (AGP). In the context of the inspection planning for a mobile robot, special attention have to be care to adequately model sensing capabilities of the real robot. The most straightforward sensing model is based on unrestricted omnidirectional visibility of a point guard, which is also assumed in the original AGP formulation. A brief overview of real sensors is presented in the next paragraphs to discussed applicability of these assumptions in the context of the inspection task in a search and rescue mission and currently used sensors.

Two types of sensors are widely used in the mobile robotics, laser based range finders and cameras [203]. The sensing capabilities of these sensors are limited, and two parameters are typically considered: the range distance and the field of view. For example the laser measurement system SICK LMS 200 provides measurement range up to 80 meters with the angle 180° [2] or SICK S300, which provides wider angle 270°, but with worse resolution [4]. The HOKUYO UTM-30LX device has 30 meters sensing range and the wide angle 270°[5] or the URG-04LX model with the maximal range 4 meters and the angle 240°. The range 80 meters can be considered practically unrestricted for typical indoor environment. However these scanners do not provide omnidirectional view, two such devices can be used in specific configurations to provide full 360° view, e.g. Figure 4.1a. Another options are to use rotating mechanism or a full 3D scanner that provide 360° horizontal field of view [117] or commercially available IMAGER 5003 (Lara 25200), which provides 360°×310° field of view, shown in Figure 4.1b. An overview of used configurations of scanners can be found in [203]. Minimum sensing range is not typically an issue, as it is small and it is negligible in comparison to size of the robot and position of the device at the robot.

The visibility range of camera depends on the used optical set, resolution and character of finding object that requires sufficient resolution to reliably recognize (inspect) an object of interest. The omnidirectional field of view can be obtained by turning camera or whole robot at the sensing location. The camera can be affected by vibrations during motion, therefore to obtain an accurate image it can be necessary to slowdown or completely stop the robot. The full angle of view can be supported by a hyperbolic mirror [249], see Figure 4.1c, or a panoramatic camera can be used, e.g SONY RPU-C251 in Figure 4.1d. The showed camera provides image with 1.3M-pixel resolution at 7.5 frame rate or 2M-pixel in case of the RPU-C2512 model [7].

(a) (b) (c) (d)

Figure 4.1: Omnidirectional sensors: (**a**) configuration of two laser scanners mounted on G^{2}Bot platform used in the Gerstner Laboratory to provide omnidirectional sensing, (**b**) 3D scanner IMAGER 5003, (**c**) Pulnix digital camera with a hyperbolic mirror, (**d**) desktop panoramatic camera SONY RPU-C251.

The examples of sensing devices show that the omnidirectional view can be realized with two laser scanners or panoramatic cameras, thus it is not an issue. On the other hand, the restricted visibility range have to be considered, mostly due to the fact, that to recognize searched object it is necessary to have sufficient level of details. Also the visibility range depends not only on capabilities of device, but also on the current environment conditions, e.g. smoke in a burning building during search and rescue mission. The AGP approaches typically do not considered particular visibility constraints, which is the main reason why authors of [107] call the problem *sensor placement* rather than the AGP.

Even in a case that only visibility range is restricted and the sensor placement is more close to the AGP, an important aspect related to the mobile robotics have to be considered. The position of the sensing location must respect size of the robot and particular surrounding environment, that means a sensing location must be reachable by the robot. The robot has to visit all necessary sensing locations on an inspection path in order to "see" the whole environment and find an object of interest. The multi-goal path planning problem can be formulated as the TSP or the MTSP, where sensing locations become cities, but path between locations have to be found. It is also important that these paths have to be traversable by the particular mobile robot. Regarding the path planning part of the inspection, a set of sensing locations should be found with respect to capabilities of particular mobile robot.

Two important aspects of the finding sensing locations have to noted. Even that optimal and polynomial AGP algorithms have been proposed for restricted class of the environment (e.g. orthogonal polygons), the problem is NP-hard for a polygon with holes, therefore only approximate solution can be expected. Also it should be noted that the AGP formulation, which minimize number of guards (cities), does not necessary lead to the shortest inspection path, because the path planning is not a part of the problem.

The rest of the chapter is organized as follows. The next section summarize problem specification. A brief summary of supporting algorithms and discussion of issues of the polygonal representation are presented in Section 4.2. Section 4.3 presents three sensor placement algorithms for restricted visibility range. Experimental results are presented in Section 4.4. The chapter is concluded with discussion of the presented results and the proposed algorithm, Section 4.5.

4.1 Problem Specification

The problem is to find a set of sensing locations that are placed in $\mathcal{C}_{free}$ of particular mobile robot and the whole environment is inspected by performing measurements at each sensing location. The robot working environment is a prior known and it is represented as a polygon with holes $\mathcal{W}$. A robot is assumed to be differential nonholonomic drive robot with an omnidirectional sensing capability. The robot is modeled by a disk with the radius ρ, thus $\mathcal{C}_{free}$is two dimensional and it can be represented by a polygonal domain. The visibility range is modeled as a disk with the radius d, where d is restricted visibility range, or a value higher than size of the $\mathcal{W}$ in the case of unrestricted visibility range. A disk is a polygon with the particular number of vertices.

Sensor Placement Problem - For a given workspace $\mathcal{W} \subset \mathbb{R}^2$, find a set of sensing locations $\boldsymbol{G}$ such that every point of $\mathcal{W}$ is d-visible from at least one point of $\boldsymbol{G}$.

The polygon representing the robot workspace is called *map* and it consists of the *border* polygon and set of *obstacle* polygons. Obstacles are polygonal representation of $\mathcal{C}_{obs}$. The polygon is formed by the ordered sequence of vertices, which define orientation of the polygon. A map edge connects two consecutive polygon vertices and it has free space of the environment on its left side, hence the orientation of the border is counterclockwise (CCW) and obstacles are clockwise (CW) oriented. Polygons are simple and do not contain three collinear vertices[1]. The map border and obstacles do not intersect even at polygon vertex. The free space is one connected component, otherwise disconnected components represent a set of polygonal maps. Without loss of generality only single component is assumed.

To follow a notation of the AGP, a sensing location is also called a guard in this thesis.

Once guards are found, the inspection planning is formulated as the routing problem to find a sequence of guards visits such that the total length of the path is minimized. Shortest paths between guards can be found as the shortest-path roadmap constructed from the visibility graph, e.g. in $O((n_v + n_g)^2)$ [210], where n_v denotes the number of map vertices and n_g is the number of guards. The path planning problem is formulated as the TSP on a graph $G(V, E)$, where V denotes guards and E is a set of edges with cost derived from the length of the shortest path between guards, the TSP can be then solved by a TSP solver. Without loss of generality $G(V, E)$ is assumed to be complete.

The quality of found set of guards is evaluated according to two costs: the cost of sensing (the number of guards) and the cost of motion (the length of the inspection path).

4.2 Supporting Algorithms

4.2.1 Environment Representation

The polygonal representation of the robot workspace can be stored in various ways. Three commonly used representations are: sequence of vertices, Double Connected Edge List (DCEL), Nef polyhedra. Each representation has particular advantage. The sequence of vertices is easy to manipulate. The main advantage of the DCEL [217] is direct access to the topological relation, however it is little bit complicated to be maintained than just a

[1]This assumption is not restrictive, because it practically means simpler and more readable algorithms, because collinearities can be often solved in the cost of more complex algorithm.

sequence of vertices. A planar Nef polygon can be obtained from a finite set of open half spaces by Boolean operations: unions, intersection, and complement [114]. Such representation can be used for planar case and it is also suitable for higher dimensions.

Operations with geometric primitives need special attention regarding the numeric precision and degenerative cases [230]. Geometric operations need precise arithmetic operations, and to avoid possible issues with limited floating-point precision, an exact representation can be used. The cost of precise representation such as GMP [111] is in increased computational time [95]. An alternative to exact representations can be extensions of the standard floating point representation that provide correct rounding. The library MPFR is an extension of the IEEE 754 that provides efficient multiple-precision floating-point arithmetic, correct rounding of implemented operations and mathematical functions [97].

For practical applications it is always necessary to select between sufficient robustness and speed of used algorithms. The selection can be based on usage of an appropriate numeric type, e.g. integer, double or arbitrary precision types. For an example the Computational Geometry Algorithms Library (CGAL) provides geometric primitives grouped into *Kernel* concept, which allows to select a required numeric type [6].

4.2.2 Polygon Filtering

However a geometric representation is memory efficient in comparison to the grid (spatial) representation, the number of vertices can be still unnecessarily high. The number of vertices of the map affects the required computational time the algorithms and also it has influence to numerical issues. For example if two vertices of a non-convex polygon are very close, removing one of them will lead to have convex polygon, which can be easily covered by one guard placing at any position inside the polygon. These reasons lead to pre-process polygonal representation of the environment to reduce the number of vertices. Such techniques are used in mapping algorithms to create a polygonal representation of surrounding environment of a mobile robot [164].

Algorithm 1: Polygon filtering based on the relevance measure

Input: $\boldsymbol{P}$ - simple polygon
Input: k - *minimal allowed value of relevance*
Result: filtered polygon $\boldsymbol{P}$
Require: $|P| > 3$

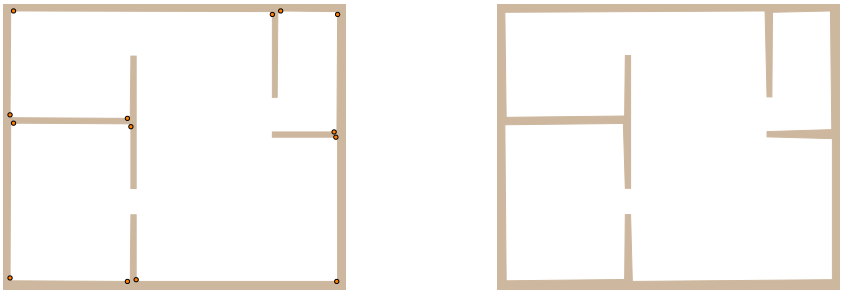
repeat
  $v_i \leftarrow \text{argmin}_{v \in \boldsymbol{P}} K(v)$
  $k_m \leftarrow K(v_i)$
  if $k_m < k$ **then**
    $\boldsymbol{P} \leftarrow \boldsymbol{P} \setminus v_i$
until $k_m \geq k$ *OR* $|P| = 3$

A polygon filter technique based on the relevance measure [171] is one of the suitable algorithm to filter unnecessary vertices, see Algorithm 1, in which $K(v)$ is the relevance measure based on the Euclidean distance

$$K(v_i) = |v_{i-1}, v_i| + |v_i, v_{i+1}| - |v_{i-1}, v_{i+1}|, \tag{4.1}$$

where v_{i-1} and v_{i+1} are neighbouring vertices of the vertex v_i in the polygon $\boldsymbol{P}$ [270].

An example of filtering is shown in Figure 4.2. The left figure shows original map with 14 highlighted filtered vertices for k=5 cm, the right figure shows final map with 23 vertices. Filtered vertices are result of automatic process to obtain polygonal representation



(a) 37 vertices (b) 23 vertices

Figure 4.2: Filtered vertices at corners of the rooms.

from a drawing of building plan. The plan is firstly scanned into raster image, then a set of morphological operations are applied, and finally pixels are approximated by line segments that are connected to form closed polygons [158].

4.2.3 Obstacle Growing

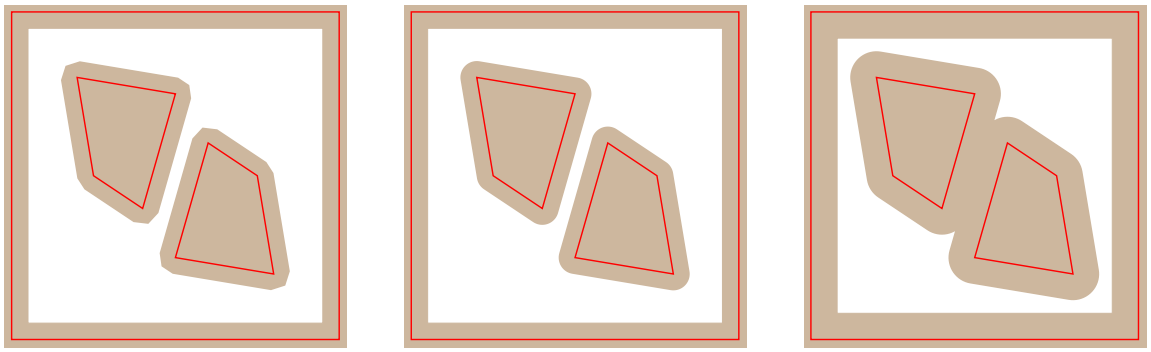
A found set of sensing locations represents positions in the environment, where a mobile robot performs measurements. To ensure that these locations are reachable by the robot, it is necessary to consider size of the robot during guards positioning. Guards have to be in $\mathcal{C}_{free}$ of the mobile robot. A polygonal representation of $\mathcal{C}_{free}$ can be found by the application of the Minkowski sum to all obstacle regions $\mathcal{O}$ and a disk $\mathcal{D}$ representing rigid body of the robot with circle circumference. The Minkowski sum can be defined as

$$\mathcal{O} \oplus \mathcal{D} = \{x + y \mid x \in \mathcal{O}, y \in \mathcal{D}\}.$$

Obstacles are expanded by the radius of the disk, while the border is shrunk. The shrunk operation represents the Minkowski difference and can be defined as $\mathcal{O} \ominus \mathcal{D} = \mathcal{O} \oplus (-\mathcal{D})$ [176]. For convex polygons the Minkowski sum can be computed exactly [94]. An approximation of the Minkowski sum or difference can be found as a buffer operation in GIS oriented computational libraries like the JTS [8] or GEOS [10].

The robot workspace and also $\mathcal{C}_{free}$ are represented by polygons, the term map denoted to the polygonal representation of the environment is also used for polygonal representation of $\mathcal{C}_{free}$ in this thesis. The main advantage of the shrunk workspace ($\mathcal{C}_{free}$) is that a point robot can be assumed. Each sensing location must be in $\mathcal{C}_{free}$, while the required guarded space is the workspace itself. For simplicity it is assumed that coverage the polygonal map representing $\mathcal{C}_{free}$ will also cover the workspace, thus only the shrunk map is considered in sensor placement algorithms.

Examples of expanded obstacles are shown in Figure 4.3, the original contour of the map is represented by the red line segments. The first two images show shrunk maps by the same value, the difference is in the approximation of the disk. The first case (Figure 4.3a) uses only two vertices at a corner, while the second (Figure 4.3b) uses 16 vertices. The rightmost contains joined obstacles into one polygonal region as a result of application of larger disk.



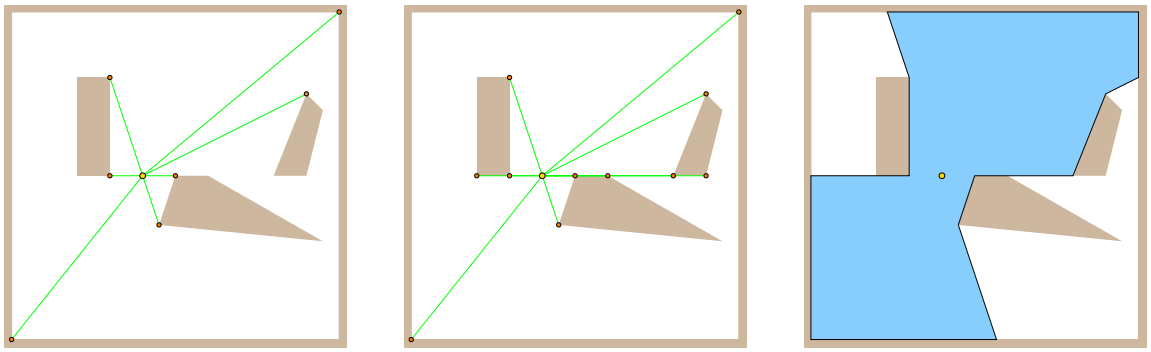
(a) (b) (c)

Figure 4.3: Example of shrunk free space, the original map is shown in red.

4.2.4 Computation of Visibility

The key component for guarding art galleries is an algorithm to find a visibility graph and a visibility polygon. If a map contains n vertices all visible vertices (or visibility polygon) for particular point can be found in $O(n \log n)$ [71]. Alternatively the full visibility graph of the map can be found in $O(n^2)$ by the algorithm based on rotation trees for a set of segments [210]. Two particular issues of the visibility computation should be considered. The first one is more related to collinearities and the second issue relates to the model of restricted visibility.

Although a map does not contain three consecutive collinear vertices, the computation of the visibility has to deal with collinearities. An example is shown in Figure 4.4, where a point p (visualized as a yellow disk) lies at a diagonal of the map. Probably the most natural way to compute visible vertices from p is shown in Figure 4.4a, while a set of visible vertices suitable for a k-searcher model of the guard is shown in Figure 4.4b. The rightmost figure shows star shaped polygon, which represents visibility polygon from p.



(a) (b) (c)

Figure 4.4: Visible vertices from the point and its visibility polygon.

A disk is used to model the restricted visibility range in a polygonal domain. A discrete representation of the disk can be formed by particular number of vertices. The area of the disk depends on the number of vertices. If the area of the visibility polygon is used in a covering algorithm, the approximation of the disk can affect the algorithm performance. Examples of several visibility polygons (disks) are shown in Figure 4.5. If only eight ver-

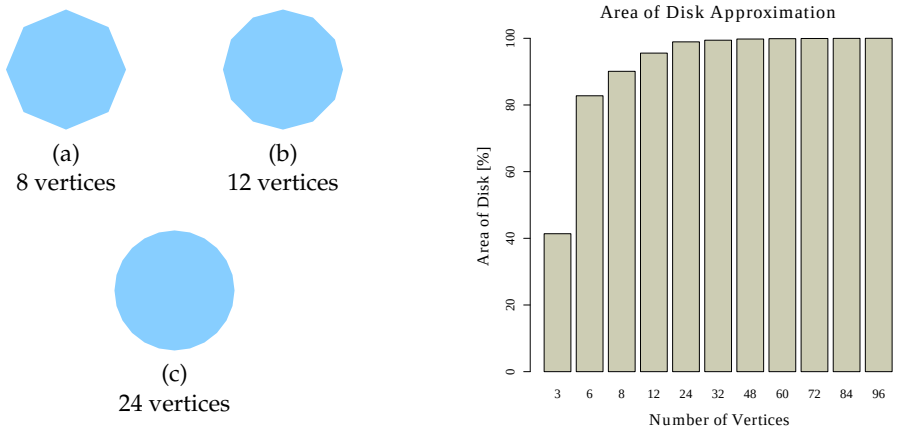


Figure 4.5: Discrete disk approximation and area of the approximation with the radius 1 m in percents of the disk area.

tices are used, the error of coverage area can be up to ten percents, which is of course only hypothetical case, because the shape of the disk always leads to use more guards for coverage rectangular-like office environments. More than 24 vertices seems be unnecessary and such approximation only increases computational burden. The disk used for the restricted visibility range has always 24 vertices in this thesis.

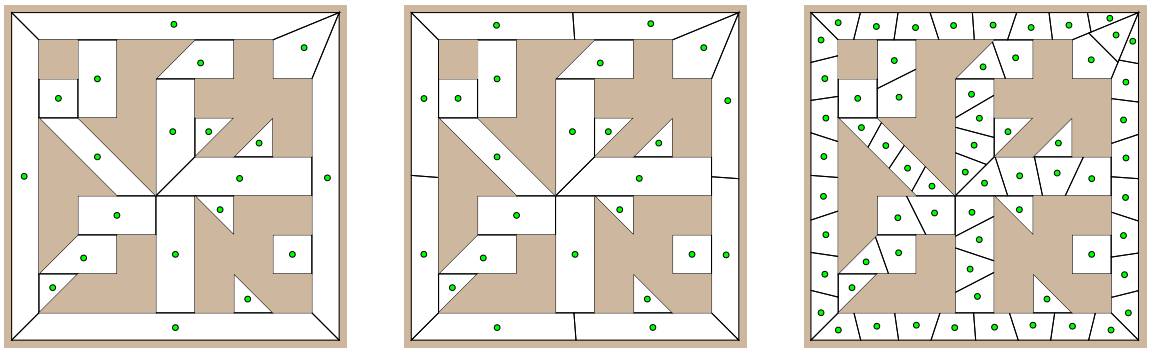
4.3 AGP - Algorithms

The simple solution to solve the AGP can be directly based on Fisk's proof [93], which utilizes triangulation of a polygon. Beside this simple solution, several AGP algorithms providing significantly less number of guards can be found in literature. Two sensor placement algorithm from the robotic domain have been selected to study the problem of finding sensing locations. The approaches have been selected, because both algorithms have been applied in mobile robotic navigation and both address the restricted visibility range constraint. The first approach is based on deterministic polygonal division [147], while the second is based on randomized sampling [108]. These two algorithms are briefly described in the next two sections. A new proposed algorithm called the *Boundary Placement* is described in Section 4.3.3, it has been designed especially for small visibility range (units of meters) and with consideration of the consecutive path planning problem.

4.3.1 Convex Polygon Partitioning - CPP

A deterministic sensor placement algorithm based on the decomposition of a polygonal environment representation into a set of convex polygons has been proposed in [147]. Each convex polygon is covered by one guard and to satisfy restricted visibility range constraint a distance from the guard to a vertex of the guarded polygon has to be less than visibility range d. If a convex polygon is too large, it is divided into convex sub-polygons until each sub-polygon can be covered by one guard with d-visibility and omnidirectional view. The primal convex partition is found by Seidel's algorithm [233] and the total complexity is linear with the number of found guards [147]. An abbreviation CPP (Convex Polygon Partitioning) is used as a reference to the algorithm in this thesis.

The CPP algorithm is deterministic and found set of guards is a property of the polygonal representation of the environment. The number of found guards depends on the partitioning to convex polygons. For a polygon with very small segments, vertices can cause additional convex polygons, which must be covered by an additional guard. This issue can be partially solved by the polygon filter technique described in Section 4.2.2. Examples of found set of guards and particular convex sub-polygons are shown in Figure 4.6.



(a) unrestricted visibility range (b) d=2.0 m (c) d=0.5 m

Figure 4.6: Found guards by the CPP algorithm for the visibility range d.

4.3.2 Randomized Dual Sampling Schema - RDS

The idea of the Randomized Dual Sampling Schema (RDS) is based on sampling the constraints of the problem (the points to be covered) instead of its domain [108]. The algorithm finds set of guards to cover the boundary of the free space and it performs in two steps. At first, the boundary is sampled by a point and its visibility polygon is computed. After that, the polygon is sampled m times and a point with the highest coverage is denoted as new guard. The visibility polygon of the new guard is subtracted from uncovered free space and the process is repeated until the whole free space is covered.

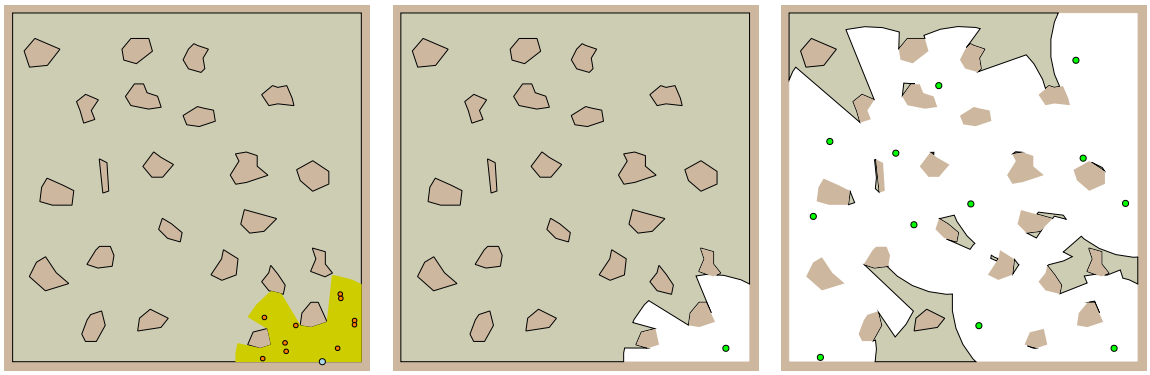
Algorithm 2: Randomized dual sampling schema

Input: $\mathcal{W}$- workspace to be covered
Input: d - maximal sensing range
Input: m - number of samples
Result: $\boldsymbol{G}$ - set of found guards (sensing locations)

$\boldsymbol{U} \leftarrow \mathcal{W}$ `// set uncovered free space`
while $|\boldsymbol{U}| > 0$ **do**
  $p_b \leftarrow$ *select random point at border of* $\boldsymbol{U}$
  $\boldsymbol{V} \leftarrow \text{visible}(\mathcal{W}, p_b, d)$ `// d-visible polygon from` p_b
  $\{p_1, p_2, \ldots, p_m\} \leftarrow$ *random points in* $\boldsymbol{V}$
  $p^\star \leftarrow \arg\max_{p_i \in \{p_1, p_2, \ldots, p_m\}} |\boldsymbol{U} \cap \text{visible}(\mathcal{W}, p_i, d)|$
  $\boldsymbol{G} \leftarrow \boldsymbol{G} \cup \{p^\star\}$
  $\boldsymbol{U} \leftarrow \boldsymbol{U} \setminus \text{visible}(\mathcal{W}, p^\star, d)$

The sensor placement procedure is summarized in Algorithm 2, where $|.|$ denotes area of the particular polygonal part of the workspace. The algorithm has been slightly modi-

fied to address coverage of the whole interior of the workspace $\mathcal{W}$ and not only its boundary. The set $\boldsymbol{U}$ represents uncovered part of $\mathcal{W}$ instead of the boundary of $\mathcal{W}$ and $|.|$ respects the interior instead of the total length of the uncovered boundary. The algorithm is complete and it is terminated after finite number of iterations, because at each iteration a random point is generated at border of $\boldsymbol{U}$ and visibility polygon of the new guard is subtracted from $\boldsymbol{U}$. An example of the algorithm performance is shown in Figure 4.7, small disks denote random points.



(a) (b) (c)

Figure 4.7: Example of the RDS performance; (**a**) initial random point p_b (in blue) at border of uncovered free space $\boldsymbol{U}$, its restricted visibility polygon $\boldsymbol{V}$ (yellow) and set of random points, (**c**) reduced uncovered free space $\boldsymbol{U}$ after the first guard has been found, (**d**) a covered free space after several iterations.

The complexity of the algorithm depends on computation of the visibility polygon, which can be done in $O(n \log n)$, where n is the number of vertices of $\mathcal{W}$. The restricted visibility is computed as an intersection of the visibility polygon and a disk. The disk is formed from k edges and its radius is the visibility range d. Newly covered portion of the free space is subtracted from U, which can increase the number of vertices up to knn_g, where n_g is the number of found guards. The overall complexity can be bounded by $O(mnn_g \log(nn_g))$, where m is the number of random samples.

An advantage of the randomized algorithm is its ability to deal with additional constraints. In [108], authors consider the minimal visibility range constraints and the incident angle constraints, which model a situation that a laser beam is not reflected to the sensing device for too wide incident angle of the laser beam with the surface normal. The angle constraint cannot be applied for the coverage of the whole free space and it is not considered in the thesis, however it can be used for the boundary cover.

4.3.3 Boundary Placement - BP

The previous algorithms provide sufficiently small number of guards (RDS) or are very fast (CPP) and they have been successfully deployed in the inspection task [108, 69, 147]. Both algorithms find a set of sensing locations independently to the consecutive path planning problem. The path can be unnecessary long even in the case of the exact TSP solution, although the motivation of the discrete sensing is higher cost of the measurement than the cost of motion to the sensing locations. The proposed *Boundary Placement* (BP) algorithm tries to consider length of the inspection path during sensor placement, which

is based on the randomization process of the RDS. The randomized sampling is guided by a priori knowledge about the environment structure and positions of already found guards. The main idea follows greedy principle and suggestion to do not place guards unnecessary far from each other, thus guards should be placed close to each other and the path to visit all guards is expected to be shorter. The idea has been experimentally verified and preliminary experiments [85] indicate better results from the length of the planned path point of view. Principle, description of the algorithm, and a new post-optimization procedure are presented in this section.

One of the most favorable location for a guard is at the reflex vertex of the map, see for an example Figure 4.3 in [211], but it is not necessary the case for the visibility with restricted range. In such case, it is needed to place guard in at least visibility range distance from an obstacle. From the path planning point of view, it is not necessary to move the robot closer to walls (or obstacles) than at a perimeter of visibility range. This consideration leads to place guards firstly at a pre-specified distance from obstacles and then place additional guards to cover the rest of the uncovered free space. The primal guards positioning is at the boundary of the shrunk free space by the particular distance. The boundary represents prior knowledge how to sample $\mathcal{W}$.

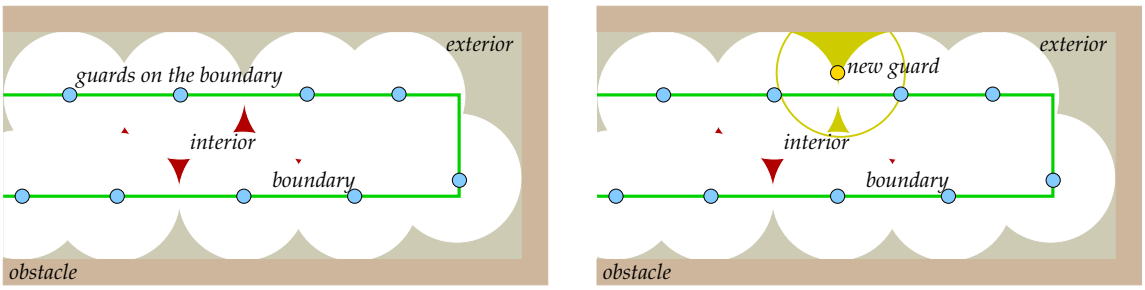
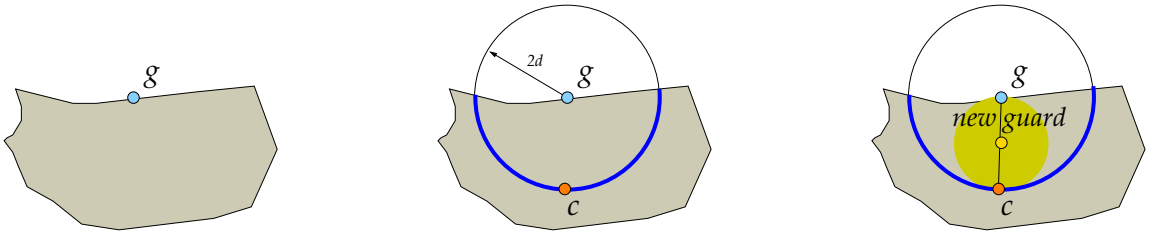


Figure 4.8: Principle of the Boundary Placement algorithm.

The main idea is demonstrated in Figure 4.8. The boundary is shown as green segments and its distance to the obstacle is very close to the restricted visibility range d. After placing guards at the boundary, the uncovered free space is divided into two sets of regions. The first set is called *interior* and it contains polygons inside the boundary, the set is represented by red polygons. The important property of these polygons is that they are not connected with obstacles. The *exterior* set is the second set and it represents regions outside the boundary. The second part of the idea can be demonstrated by an example in the same figure. If the boundary is covered by the set of blue guards, see Figure 4.8b, then to cover a part of the exterior, new guard can be place in a certain distance from the already placed guards. A path connecting such sensing locations is directed by the guards at the boundary, while added guards, like the yellow guard in the figure, do not lead to significant change of the path direction. This idealized case demonstrates the main idea behind the algorithm design.

The BP algorithm consists of four parts. The first three parts correspond to covering the boundary, interior and exterior sets. The fourth path is a post-processing procedure to reduce the number of found guards by replacing two very close guards by one guard with the same coverage. Similar randomized schema to the RDS is used, but the second sampling is replaced by two heuristic strategies.

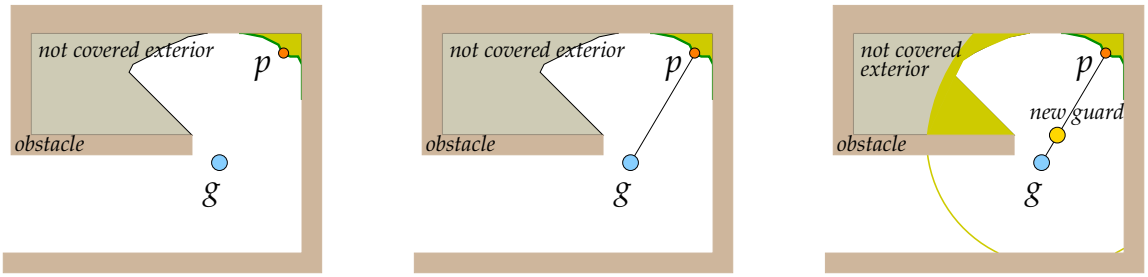
Large region cover strategy - firstly selects a random point g at the boundary of the un-



(a) (b) (c)

Figure 4.9: Large region cover strategy.

covered region, which is not a part of an obstacle, see Figure 4.9a. Then a midpoint c of the longest part of the circle (with the radius $2d$, where d is the visibility range) lying inside the uncovered region is determined, Figure 4.9b. Finally, a new guard is placed in the middle of the segment (g, c), see Figure 4.9c.



(a) a random point (b) the closest visible guard (c) new guard

Figure 4.10: Small region cover strategy.

Small region cover strategy - also starts with a random sample point p at the boundary, which is not a part of an obstacle, see Figure 4.10a. Then, the closest already found guard g, which is directly visible from p, is determined. If such a guard is not found, the point p is used as a new guard. Otherwise a new guard is placed at the segment (g, p) close to g as much as possible, see Figure 4.10c. The new guard should cover same portion of the uncovered free space as the point p. More precisely, p lies at a border of one of the region that is part of the set of regions. Only this particular region is being covered, which means only its covered area by p is considered.

The BP algorithm is summarized in Algorithm 3. The second and third parts of the algorithm are almost identical, except the set of uncovered free space $\boldsymbol{I}$ resp. $\boldsymbol{E}$. It is expected that the cover of the interior will also cover part of the exterior, that is why the covering interior precedes the covering exterior. The symbol δ denotes a border of the polygon. The polygon is an open set, thus the border is difference between the closure of the polygon and the polygon. Both the interior and exterior are collections of polygons, hence a particular polygon from the collection is denoted as $\boldsymbol{I}_p \in \boldsymbol{I}$ or $\boldsymbol{E}_p \in \boldsymbol{E}$.

The cover strategy is the *large region cover strategy* or *small region cover strategy*. It is selected according to area of the particular polygon $\boldsymbol{I}_p$ or $\boldsymbol{E}_p$,

$$|\boldsymbol{I}_p| \geq \begin{cases} \mu_I \pi d^2 & \textit{large region cover strategy,} \\ otherwise & \textit{small region cover strategy.} \end{cases} \tag{4.2}$$

An area of the visibility disk (with the radius d) is multiplied by μ_I or μ_E to estimate size of the particular uncovered region $\boldsymbol{I}_p$ or $\boldsymbol{E}_p$ respectively.

Algorithm 3: Boundary Placement Algorithm

Input: $\mathcal{W}$ - workspace to be covered
Input: d - visibility range
Input: $\delta\boldsymbol{B}$ - the boundary (border of shrunk free space)
Result: $\boldsymbol{G}$ - set of found guards (sensing locations)

$\boldsymbol{U} \leftarrow \mathcal{W}$ `// set uncovered free space`

Part I **while** $|\delta\boldsymbol{B}| > 0$ **do**
- $g \leftarrow$ *select random point at* $\delta\boldsymbol{B}$
- $\delta\boldsymbol{B} \leftarrow \delta\boldsymbol{B} \setminus \text{visible}\,(\mathcal{W}, g, d)$
- $\boldsymbol{G} \leftarrow \boldsymbol{G} \cup \{g\}$ `// add boundary guard to the set of guards`
- **Part I** - *Boundary Cover*

Part II $\boldsymbol{T} \leftarrow \boldsymbol{U} \setminus \bigcup_{g_j \in \boldsymbol{G}} \text{visible}(\mathcal{W}, d, g_j)$ `// determine uncovered free space`

$\boldsymbol{I} \leftarrow \{\boldsymbol{I}_i | \boldsymbol{I}_i \in \boldsymbol{T} \wedge (\boldsymbol{I}_i \cap \delta\mathcal{W} = 0)\}$ `// interior set`

while $|\boldsymbol{I}| > 0$ **do**
- $p \leftarrow$ *select random point at* $\delta\boldsymbol{I}, p \in \boldsymbol{I}_p, \boldsymbol{I}_p \in \boldsymbol{I}$
- $g \leftarrow \text{cover\_strategy}(p, |\boldsymbol{I}_p|)$
- $\boldsymbol{I} \leftarrow \bigcup_{\boldsymbol{I}_i \in \boldsymbol{I}} \boldsymbol{I}_i \setminus \text{visible}\,(\mathcal{W}, g, d)$ `// cover interior set`
- $\boldsymbol{G} \leftarrow \boldsymbol{G} \cup \{g\}$ `// add interior guard to the set of guards`
- **Part II** - *Interior Cover*

Part III $\boldsymbol{E} \leftarrow (\boldsymbol{U} \setminus \bigcup_{g \in \boldsymbol{G}} \text{visible}(\mathcal{W}, g, d)) \setminus \delta\mathcal{W}$ `// exterior set`

while $|\boldsymbol{E}| > 0$ **do**
- $p \leftarrow$ *select random point at* $\delta\boldsymbol{E}, p \in \boldsymbol{E}_p, \boldsymbol{E}_p \in \boldsymbol{E}$
- $g \leftarrow \text{cover\_strategy}(p, |\boldsymbol{E}_p|)$
- $\boldsymbol{E} \leftarrow \bigcup_{\boldsymbol{E}_i \in \boldsymbol{E}} \boldsymbol{E}_i \setminus \text{visible}\,(\mathcal{W}, g, d)$ `// cover exterior set`
- $\boldsymbol{G} \leftarrow \boldsymbol{G} \cup \{g\}$ `// add exterior guard to the set of guards`
- **Part III** - *Exterior Cover*

Part IV $\boldsymbol{G} \leftarrow \text{reduce}(\mathcal{W}, d, \boldsymbol{G})$ `// post-processing optimization`

The fourth part of the algorithm is a post-processing optimization to reduce the set of found guards. A guard can be placed close to previously found guards, therefore guards can cover large portion of the same space and can be possibly replaced by one guard. The optimization procedure is following.

1. Let $\mathcal{W}$ is a polygonal map and $\boldsymbol{G}$ is a set of guards.
2. Create pairs of mutually visible guards $\{G_1, \ldots, G_n\}$ that are closer than the visibility range d, $G_i = \{g_i, g_i'\}$, $g_i \neq g_i'$, $|(g_i, g_i')| \leq d, g_i \in \boldsymbol{G}, g_i' \in \boldsymbol{G}$.
3. Sort the pairs according to distance between guards and select pairs with shortest distance between guards such that each guard is only in one such pair, $\boldsymbol{G}_P = \{G_1, \ldots, G_k\}$, $g_i \in G_i, g_i \notin G_j, i \neq j, i, j \in \{1, \ldots, k\}$.
4. Compute coverage of guards, which are not in the selected pairs, the uncovered free space by these guards is denoted as $\boldsymbol{U}$.
5. Select pair G_s from $\boldsymbol{G}_P$ with the closest guards. Compute coverage of the guards $G_s = \{g_s, g_s'\}$ as $\boldsymbol{P}_s = \text{visible}(\mathcal{W}, g_s, d) \cap \boldsymbol{U}$, $\boldsymbol{P}_s' = \text{visible}(\mathcal{W}, g_s', d) \cap \boldsymbol{U}$ and coverage of the midpoint $p = \text{midpoint}(g_s, g_s')$, $\boldsymbol{P_p} = \text{visible}(\mathcal{W}, p, d) \cap \boldsymbol{U}$. Select guards according to the following criterions.
   (a) If $(\boldsymbol{P}_s \cup \boldsymbol{P}_s') \setminus \boldsymbol{P}_p = \emptyset$ then replace guards g_s, g_s' by the new guard p, $\boldsymbol{G} \leftarrow \{p\} \cup \boldsymbol{G} \setminus \{g_s, g_s'\}$ and update uncovered free space $\boldsymbol{U} \leftarrow \boldsymbol{U} \setminus \boldsymbol{P}_p$, go to step 6.

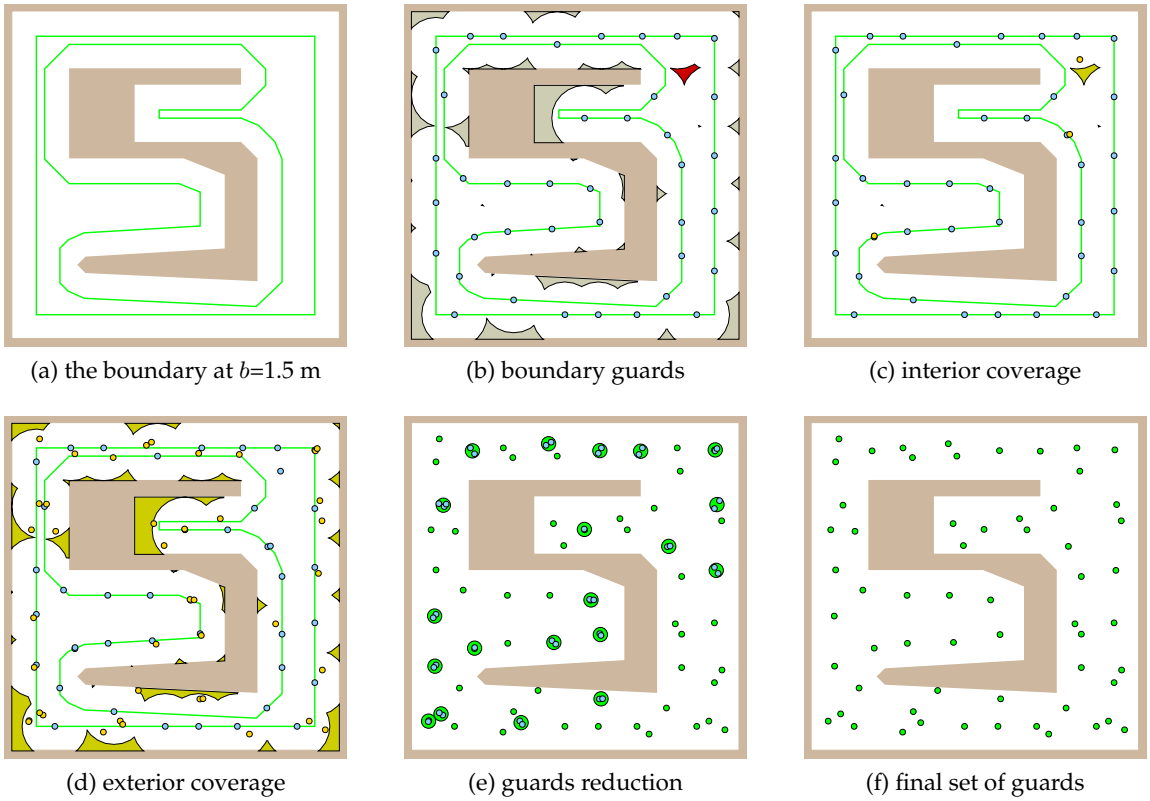
(b) If $|\boldsymbol{P}_s| > |\boldsymbol{P}'_s| \wedge \boldsymbol{P}'_s \setminus \boldsymbol{P}_s = \emptyset$ then use g_s, $\boldsymbol{G} \leftarrow \boldsymbol{G} \setminus \{g'_s\}$ and update uncovered free space $\boldsymbol{U} \leftarrow \boldsymbol{U} \setminus \boldsymbol{P}_s$, go to step 6.

(c) If $\boldsymbol{P}_s \setminus \boldsymbol{P}'_s = \emptyset$ then use g'_s, $\boldsymbol{G} \leftarrow \boldsymbol{G} \setminus \{g_s\}$ and update uncovered free space $\boldsymbol{U} \leftarrow \boldsymbol{U} \setminus \boldsymbol{P}'_s$, go to step 6.

(d) use both guards g_s and g'_s, update uncovered free space $\boldsymbol{U} \leftarrow \boldsymbol{U} \setminus (\boldsymbol{P}_s \cup \boldsymbol{P}'_s)$, go to step 6.

6. Remove the processed pair G_s from the set of pairs $\boldsymbol{G}_P \leftarrow \boldsymbol{G}_P \setminus \{G_s\}$.
7. Repeat step 5 if $\boldsymbol{G}_P$ is not empty.

An example of partial solutions in particular parts of the BP algorithm is shown in Figure 4.11.



(a) the boundary at b=1.5 m (b) boundary guards (c) interior coverage

(d) exterior coverage (e) guards reduction (f) final set of guards

Figure 4.11: An example of BP performance, visibility range d=2 m.

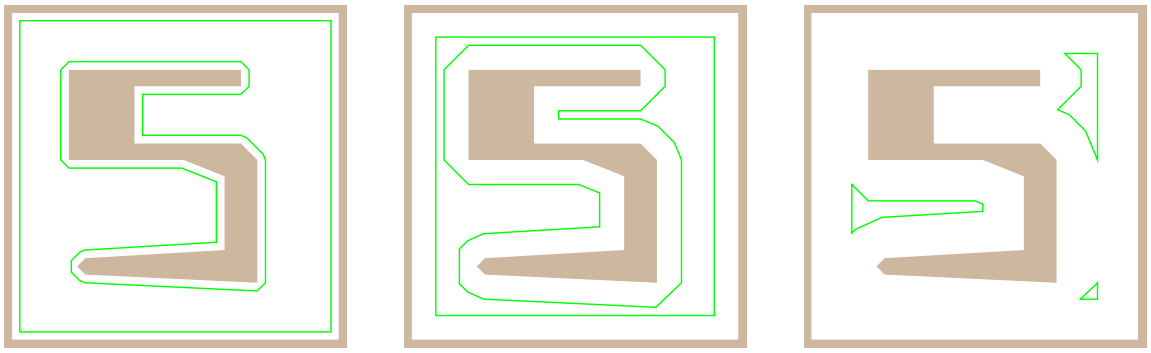
Complexity of the algorithm is very similar to the RDS. Assume $\mathcal{W}$ is represented by n vertices. The first part requires computation of visibility polygon for each new guard, which can be done in $O(n \log n)$. The second and third parts are little bit complicated as they require computation of the closest guard and determination of the new point to cover the region. The closest visible guard g_c can be found as an intersection of the full visibility region of the point p with the set of guards[2]. The point p belongs to the particular polygon of the polygon collection $\boldsymbol{I}$, or $\boldsymbol{E}$, the furthest vertex v of the particular polygon $\boldsymbol{I}_p$, or $\boldsymbol{E}_p$, is determined according to its Euclidean distance to g_c. The distance between g_c and v is used to estimate the position of the new guard g_c according to the visibility range d. Then, a visibility polygon for the new guard candidate is determined. The small cover strategy

[2]E.g. by Boolean operation on Nef polyhedra representation.

is applied for small polygons of $\boldsymbol{I}_p$ or $\boldsymbol{E}_p$, therefore its number of vertices is expected to be less than n and determination of the vertex v is negligible according to computation of required visibility polygon from p. That means time complexity can be bounded by $O(n \log(n))$. The large cover strategy also requires additional computation of a visibility polygon. At first, a midpoint of the longest circle part is determined in $O(k)$ steps, where k is the number of disk vertices, then the guard candidate is placed at the center of the segment from the random point at the border of the uncovered polygon and the midpoint. Similarly to the RDS, coverage of the new guard is subtracted from the uncovered free space, which can increase the number of vertices. The complexity of the first three parts depends on the visibility polygons and can be bounded by $O(nn_g \log(nn_g))$, where n_g is the number of found guards.

The last optimization procedure requires computation of mutually visible guards, which can be done in $O((n + n_g)^2)$. All pairs can be sorted in $O(n_g^2 \log(n_g^2))$, but only $n_g/2$ pairs can be selected at maximum and only $n_g/2$ new guards can be determined, therefore complexity of the determination of visibility polygons is not increased. The overall algorithm complexity can be bounded by $O(nn_g \log(nn_g) + n^2 + n_g^2 \log(n_g^2))$.

The performance of the BP algorithm mostly depends on the initial boundary selection that can be obtained as a border of the shrunk free space. The shrunk free space can be found by the Minkowski sum of the polygon and a convex disk, therefore the boundary determination can be bounded by $O((nk)^2)$, where n is the number of polygon vertices and k is the number of disk vertices.



(a) the boundary at 0.5 meters (b) the boundary at 1.5 meters (c) the boundary at 2.5 meters

Figure 4.12: Examples of the boundary at different distances from obstacles of the polygonal map.

The original idea of the BP algorithm expect the boundary at a distance similar to the visibility range. If a boundary is created at a very small distance to the border of the polygon, guards will be placed unnecessarily close to obstacles. On the other hand, if a distance is relatively high, the boundary created from the shrunk free space can be degenerated. In a degenerated case, a large portion of the free space is a part of the exterior and heuristic approach is used. Examples of the boundary found as border of the shrunk free space by a distance b are shown in Figure 4.12. Performance of the algorithm for various boundaries and visibility ranges has been studied and results are presented in the next section.

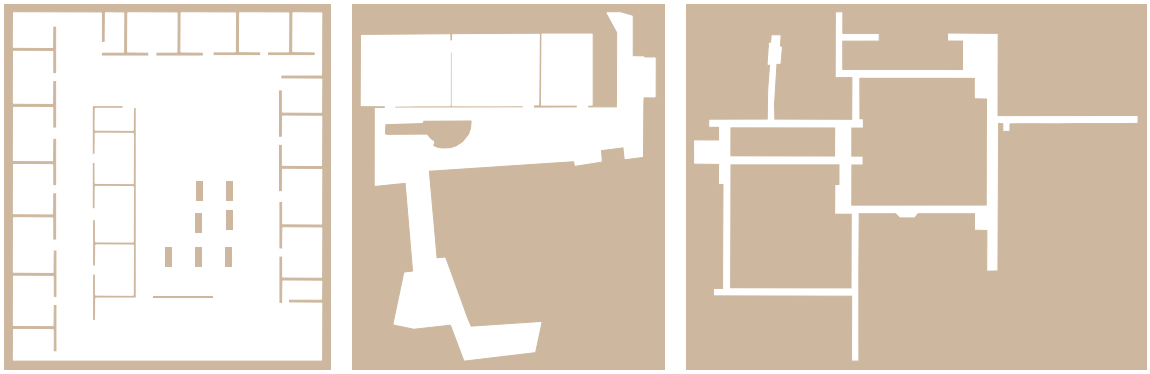
4.4 Experimental Results

The performance of the presented sensor placement algorithms have been experimentally evaluated within three maps of real environments[3]. The real environments have been selected because they provide representative experimental results. The environments were used for real experiments in a search and rescue missions during solution of the PeLoTe project. Polygonal maps of these environments have been obtained semi-automatically from the CAD models of the buildings. Properties of the maps are presented in Table 4.1.

Map name	**Size** [m × m]	**No. Holes**	**No. Vertices**
jh	20.6× 23.2	9	277
ta	40.1× 47.2	2	125
pb	133.5×105.0	3	137

Table 4.1: Basic properties of used maps.

The environments represent typical office like indoor environment, which provide realistic view to particular behaviour of studied algorithms for restricted visibility ranges. The maps are visualized in Figure 4.13. The map *jh* contains several rooms and however



(a) map *jh* 20.6 × 23.2 m (b) map *ta* 40.1 × 47.2 m (c) map *pb* 133.5 × 105.0 m

Figure 4.13: Testing environments *jh*, *ta* and *pb*.

it looks like rectangular environment, due to noisy data not all rooms are rectangular. The map *ta* is larger and contains only three rooms and wide long corridors. The main feature of this map is an obstacle inside the free space, because the shape is rounded it contains several reflex vertices, which increases difficultness to cover the free space. The last map *pb* is the largest environment and it is represented mostly by straight corridors.

At first, each algorithm is examined separately to find the most suitable set of its parameters. The experimental evaluation is performed for a set of visibility ranges $\{inf, 10.0, 5.0, 4.0, 3.0, 2.0, 1.5, 1.0\}$ meters, where *inf* denotes the unrestricted visibility range. The quality of solution is evaluated according to two costs: the number of found guards and length of the tour over the set of guards as a solution of the related TSP. The TSP is solved exactly by the Concorde solver [16] up to 600 guards (cities). For higher number of guards

[3]Besides, the algorithms have been experimentally verified in additional environments, but mainly to test the robustness without detail performance evaluation.

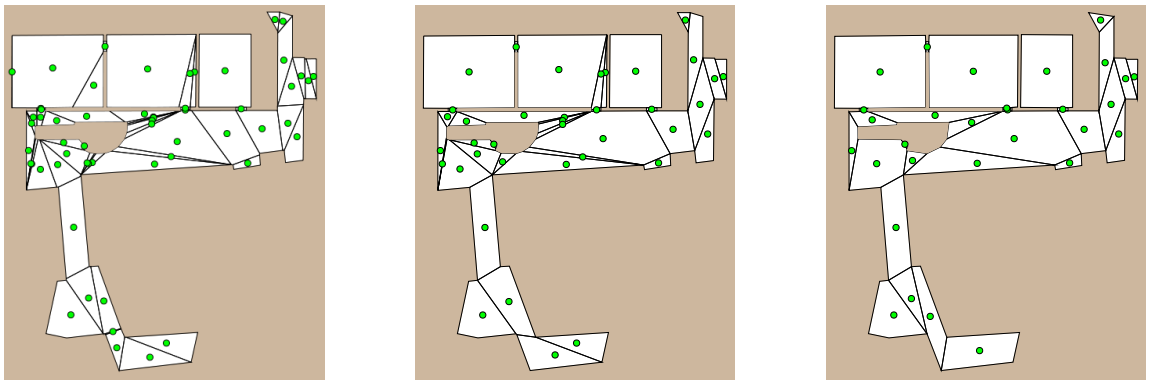
Chained Lin-Kernighan heuristic [17] is used, because required computational time to find the exact solution is too high (several hours). It practically means that almost all problems have been solved exactly for all visibility ranges and maps *jh* and *ta*, and visibility ranges higher than two meters and map *pb*.

The RDS and the BP algorithms are randomized, therefore 20 solutions are found for each problem[4]. To decrease the number of presented values, the standard deviations are not presented, they are generally in units of guards and increase with lower visibility range. Standard deviations of lengths are in units of percents and decrease for lower visibility range. Presented absolute values of the number of guards and length of the tour are rounded to integers, which is natural for the guards and tenths represents only fraction of percent of the length. To compare performance of algorithms, ratios are used according to selected algorithm variant. All computations have been performed using a single core of the Athlon X2 at 2 GHz CPU, 1 GB RAM, running FreeBSD 7.1, however algorithms have been implemented with different libraries.

4.4.1 Algorithm CPP

The CPP algorithm is sensitive to the number of map vertices, or more precisely to the number of reflex vertices. Performance of the algorithm according to the number of map vertices has been studied for maps filtered by the algorithm presented in Section 4.2.2. Five values of the relevance measure k have been considered, from one up to five centimeters. For k=5 cm the number of vertices is reduced almost two times. Higher value of k can lead to remove more vertices, but maps are too degenerative from human point of view. Also produced polygon can be non-simple and inappropriate for the CPP algorithm. Particular properties of the filtered maps are presented in Table A.1. Results for selected values of k are presented in Table A.2.

Solutions for the map *jh* does not benefit from the reduced number of vertices, because for k=0 almost all rooms are covered by one guards, see Figure A.1. Simplification of the obstacle in the map *ta* is the main reason for the significant reduction of the number of found guards, see Figure 4.14. Notice the guard in the left narrow polygon, almost at the



(a) k=0 cm (b) k=1 cm (c) k=5 cm

Figure 4.14: Found guards by the CPP, map *ta*, with unrestricted visibility range.

edge of the border, caused by an "improper" vertex that is closer than one centimeter to

[4] In a preliminary study 50 solutions have been performed and compared with results from 20 solutions, but differences are only minor.

another map vertex. The guard is not needed for the filtered map. The map *pb* contains almost only long corridors and reduction for smaller visibility ranges is not significant. Examples of found solutions are shown in Figure A.2.

Required computational time

Required computational time linearly increases with the number of found guards as authors discussed in [147], see Figure 4.15. The algorithm has been implemented in C++ and CGAL library version 3.3.1. Used geometric kernel has been *Exact_predicates_exact_constructions_kernel_with_sqrt*, which provides sufficient precision without significant computational requirements. The program has been compiled by the G++ 4.2 with -O2 optimization flag. Required computational time is in hundreds of milliseconds for less than 500 guards. The largest problem width 1852 guards has been solved in 3.3 seconds.

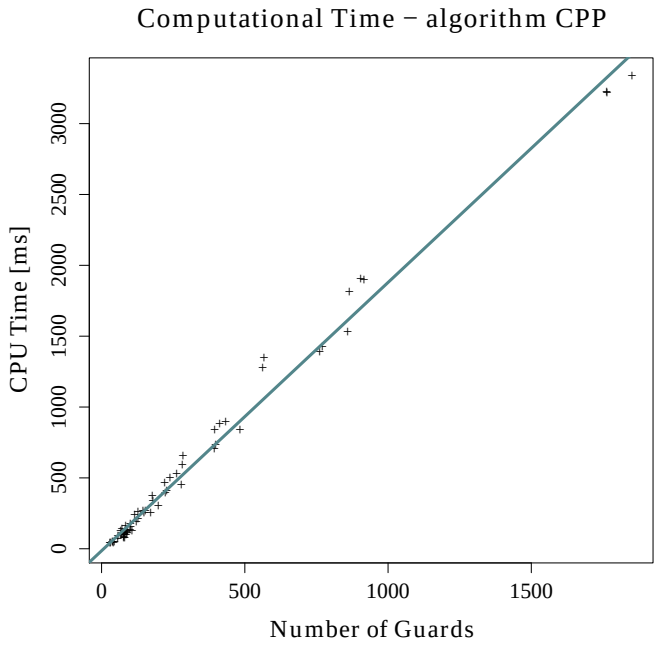


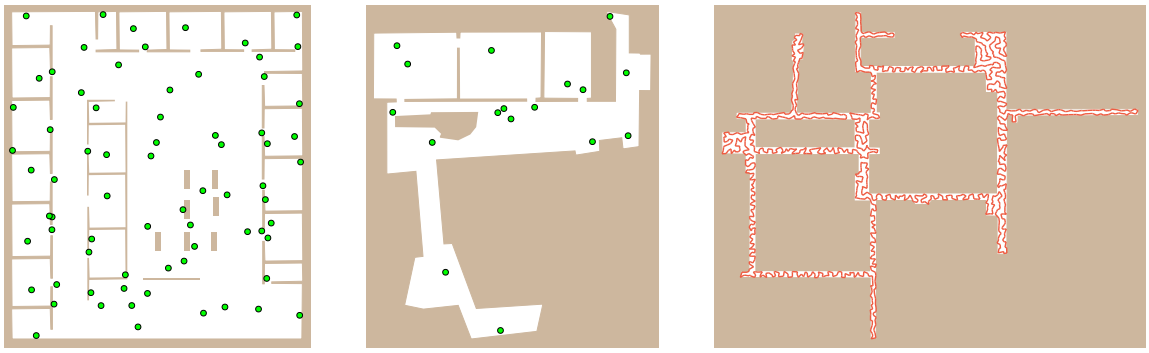
Figure 4.15: Required computational time of the CPP algorithm.

4.4.2 RDS - Randomized Dual Sampling Algorithm

The RDS algorithm has only one specific parameter the number of random samples m. The quality of found solution depends on m and one can expect less number of found guards for higher m. The most suitable value of m is selected according to trade-off between quality of solution and required computational time. The RDS has been evaluated for $m \in \{1, 5, 10, 25, 50, 75, 100\}$. The algorithm is not sensitive to the number of map vertices, it only increases required computational time, therefore filtered maps for relevance k=5 cm have been used. For each particular configuration (map, visibility range and m) 20 solutions have been found and the average number of guards and the length of the tour has been computed. The experimental setup has 168 unique configurations with 3 360 total number of found solutions. The quality of solutions according to the value of m

is compared as guards and length ratios, where reference values are for m=1, thus lower value of the ratio means better solution.

Detail results are presented in Table A.3 and selected results in Figure A.3. The highest reduction of the number of guards and length of the tour is for unrestricted visibility range. The reduction is not significant with decreasing visibility range. The most sensitive map to the number of samples is the map *jh*. For the maps *ta* and *pb* the number of guards is decreased with higher value of m, but more samples does not significantly affect the length of the tour. It is not necessary to used high number of samples for such type of maps, because quality of solutions is not increased. Examples of found solutions are presented in Figure 4.16.



(a) map *jh*, d=3 m, m=100

(b) map *ta*, d=10 m, m=25

(c) map *pb*, d=1 m, m=10, a tour over found set of guards

Figure 4.16: Example of solutions found by the RDS algorithm for the visibility range d and the number of random samples m.

To select a single value of m the overall average ratios can be handful, see Table 4.2. Regarding the presented results parameter m=25 provides the best trade-off between the solution quality and required computational time.

n	**Guards Ratio**	**Length Ratio**
1	1.00	1.00
5	0.88	0.95
10	0.87	0.94
25	0.87	0.93
50	0.87	0.93
75	0.88	0.93
100	0.89	0.93

Table 4.2: Overall ratios according to the number of random samples m.

Required Computational Time

First of all it must be noted that computational requirements are substantially affected by particular implementations. However the worst case complexity of the algorithm is used as a performance indicator, real computational requirements of operations with geometric primitives depends on the used numeric precision model. Also the computational

time is affected by the used compiler and runtime environment. The RDS algorithm has been implemented in Java with geometric library JTS [8] and all operations have been performed in the double IEEE 754 precision. The used java runtime machine has been diablo-jdk1.6.0 [9]. The computational time has been measured within java runtime environment.

The required computational time depends on the visibility range and particular map, but it is always related to the total number of found guards, therefore the required computational time is shown as histogram for selected sets of found guards in Figure 4.17. The smallest problems are solved in tens or hundreds of milliseconds. The largest problem with 1291 guards is solved in 84 seconds for m=25, while it takes more than three hundreds seconds for m=100.

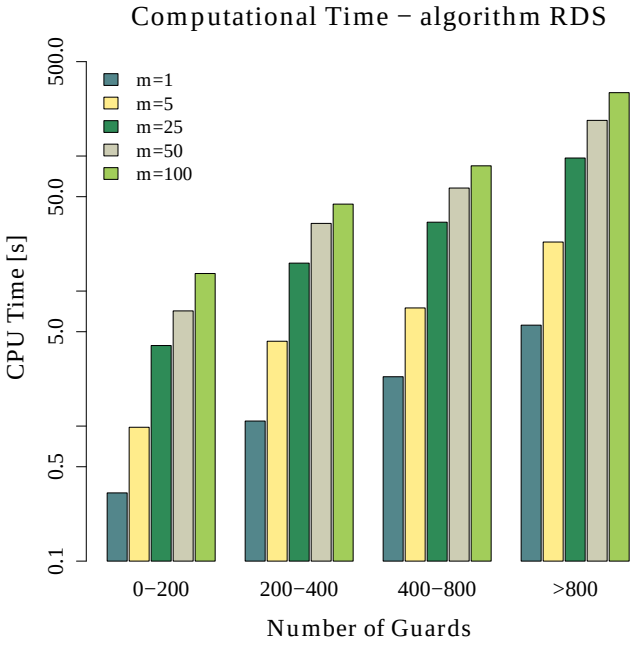


Figure 4.17: Required computational time of the RDS algorithm.

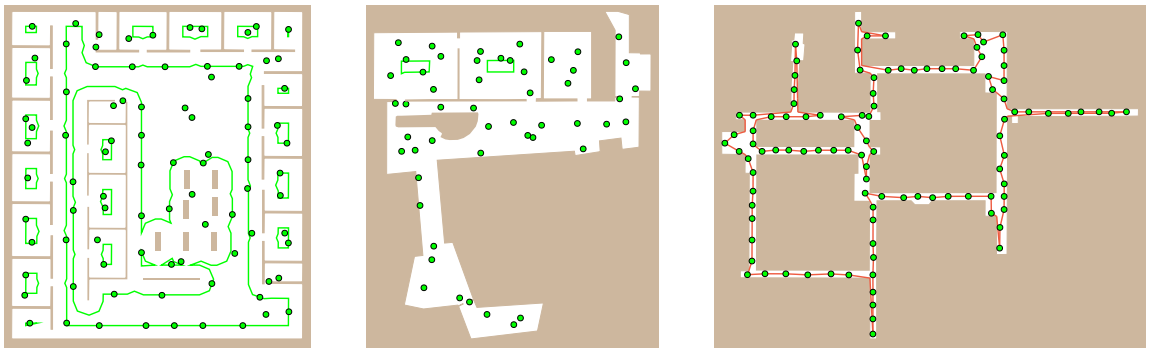
The used java runtime environment provides interesting observation. For each configuration a batch of 20 iterations has been run in a single process execution. For lower numbers of samples and high visibility ranges the first iteration has been more than three times slower than others. This behaviour relates with the runtime optimization and has been observed only for solutions found in hundreds of milliseconds. Beside the curiosity, it also indicates potential issues in the comparison of the real computational times.

From the numerical stability point of view a great advantage of the RDS algorithm is its incremental computation. If a polygonal operation like difference of new covered area from the free space or an intersection of the visibility polygon with a disk fails due to limited numeric precision, then new random point can be generated. The process can iterate until new random point allows correct execution of the operation. The total number of recorded fails has been 3867, which represents 0.560 % of the total number of found guards or 0.015 % of the total number of performed guards selections. The most problematic map is the *jh* with 2120 fails. During solving problems in the map *ta* 1378 fails has been recorded.

4.4.3 BP - Boundary Placement Algorithm

The most important parameter of the BP algorithm is the boundary $\delta\boldsymbol{B}$, and therefore the performance of the BP algorithm is studied for various $\delta\boldsymbol{B}$. Additional parameters are μ_I and μ_E, which are used to select between the large and the small region cover strategies in the interior and exterior parts of the algorithm. Both parameters have been set to the same value 0.66.

The boundary $\delta\boldsymbol{B}$ is created from the shrunk free space $\mathcal{W}$, and the distance of the boundary from the obstacles is denoted as b - the boundary value. The value of b is selected individually for each examined map, particularly the maps *jh* and *pb* are examined with boundary values 1.0, 1.5 and 2.0 meters and the map *ta* with values 1.0, 2.0, 3.0, 3.5 and 4.0 meters. For each particular configuration 20 solutions are found and average values of quality metrics are determined, similarly to the examination of the RDS algorithm. The selected results are presented in Figure A.4. The value of b affects the number of found guards in each part of the BP algorithm, see Figure A.5. For small values and high visibility ranges almost whole free space is covered by the boundary guards. In contrary, high values of b lead to very small shrunk free space and almost all free space have to be covered by guards inside the exterior. Examples of found solutions are shown in Figure 4.18.



(a) map *jh*, d=1 m, b=1 m (b) map *ta*, d=4 m, b=4 m (c) map *pb*, d=4 m, b=1 m

Figure 4.18: Example of solutions found by the BP algorithm for visibility range d and the boundary value b, green lines represent the boundary; (**a**) 88 found guards in the map *jh*, (**b**) found guards in the map *ta*, (**c**) found guards and the inspection path in the map *pb*.

An appropriate value of b should be set according to the environment and the visibility range. The map *jh* is almost always covered by guards on the boundary (for b=1 m) or guards found in the exterior part (for b=2 m). For environment with large free space, like the map *ta*, the number of boundary guards and the interior and the exterior guards are almost equal. The boundary part of the algorithm is mostly utilized for b=1 m, in other cases most guards are found in the exterior. For higher visibility ranges it is better to used boundary farther from the obstacles and for small visibility ranges the best is to use boundary at the same distance as the visibility range. The boundary should sufficiently describe the shape of the environment, but even with degenerated cased, like b=4 m for the map *ta*, solutions found by the large and small region cover strategies are not worse than 2% from the length of the tour point of view.

A selection of an appropriate b should also take into account the required computational time, and therefore the most computationally intensive part of the BP algorithm

may be considered. The most time expensive procedure is the small region cover strategy, which is mostly utilized in the exterior part. Particular spent time in each part of the BP algorithm is shown in Figure A.6. Even in case that more than 80% of guards are found at the boundary, more than 40% of computational time is spent in the exterior part. The post-optimization procedure takes from 10% to 20% percents of the computational time.

The selected values of b are presented in Table 4.3. For small visibility range the boundary is in one meter from obstacles for all used maps, while higher visibility ranges require particular value of b for each map to obtain better results. These values do not provide the smallest number of found guards, but they have been selected to minimize the number of parameters and the length of the tour.

Map	**Boundary Distance** b [m]	
	Visibility Range ≤ 2 m	**Visibility Range** > 2 m
jh	1.0	1.5
ta	1.0	3.0
pb	1.0	2.0

Table 4.3: Selected values for the boundary parameter b of the BP algorithm.

The efficiency of the post-optimization procedure to reduce the set of guards is demonstrated in Figure 4.19, where selected overall ratios are visualized, the reference value is particular quality metric before the reduction. The procedure decreases the number of guards about ten percent, while the length of the tour is almost identical.

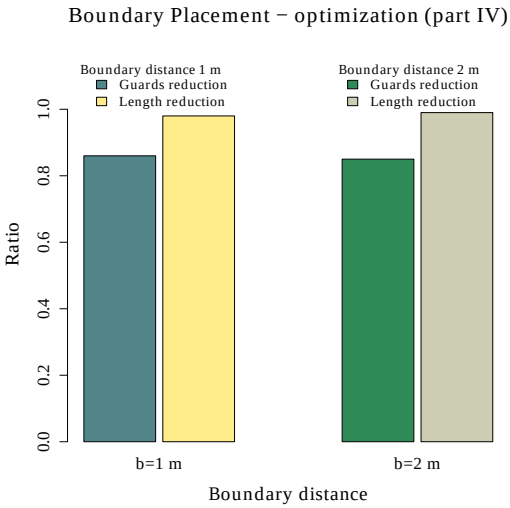


Figure 4.19: Quality of solution after fourth part of the BP algorithm.

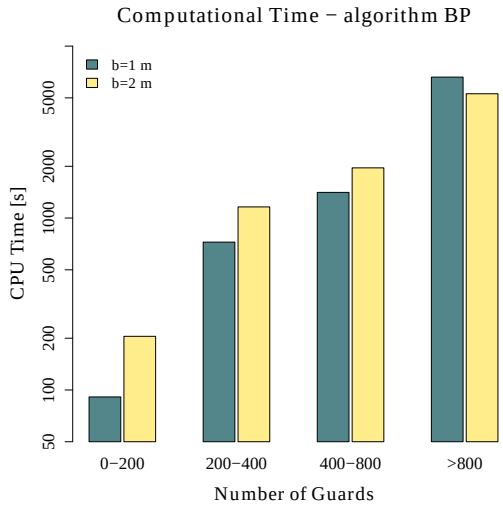


Figure 4.20: Required computational time of the BP algorithm.

Required Computational Time

The required computational time of the BP algorithm is mostly affected by the used geometric kernel of the CGAL library version 3.1. The covered free space is represented as Planar Nef polyhedra, which support open and close boundary of particular polygon. It allows determination of part of the boundary, which belongs to the border of the

free space. Polygon operations are very slow, due to creation of new points as a result of polygons difference, the precise rational representation (`CGAL::Gmpz`) leads to very large values of nominator and denominator. The usage of selected precise geometric kernel has been necessary, because it is required by the Nef polyhedra structures and also less accurate kernels leads to topological assertions during polygonal operations.

To speed up the performance and to get particular solution in a reasonable time (less than two hours) following trick has been applied. After several algorithm steps, current polygonal representations have been rounded to decrease computational burden. The rounding can lead to place points inside of obstacles, which is not allowed for computation of visibility, therefore the free space has been shrunk by a small value, at the beginning of the computation. The algorithm covers this shrunk free space, while visibilities are computed in the original (larger) map. The error of coverage is negligible and it can always be decreased by enlarging the map, because rounding is performed to one floating point value, which represents units of centimeters. After applying this implementation trick, the solution is found in tens of seconds for high visibility ranges and up to 110 minutes for the visibility range one meter and the map *pb*. The program has been compiled by the G++ 3.4.6 with -O2 optimization flag. Required computational times are shown in Figure 4.20. The presented required times does not include computation of the boundary. The boundary is a shrunk free space and its creation requires tens of milliseconds, hence it is negligible according to the required time of the BP algorithm.

4.4.4 Algorithm Comparison

Performance of all algorithms has been examined according to identified particular algorithm settings. To provide the most suitable maps for each algorithm, the maps have been modified as follows. To support rounding technique in the BP algorithm, free space of each map has been shrunk by very small value (units of centimeters). Then the shrunk free space has been filtered with k=5 cm to support the CPP algorithm. These filtered free spaces represent maps to be covered by the all algorithms. The number of random samples m of the RDS algorithm has been set to 25 and the boundary values for the BP have been selected according to Table 4.3. Similarly to the previous experiments 20 solutions are found for each particular configuration and average values are determined. Ratios are computed according to solution found by the CPP, the guards ratio is denoted as GR and the length ratio as LR, and the sample standard deviations are denoted as s_{GR} and s_{LR}. Overall results for all examined maps (*jh*, *ta* and *pb*) are shown in Table 4.4, detail results are presented in Table A.4 and Figure A.7.

Both randomized algorithms (the RDS and the BP) outperforms the CPP in the number of guards as well as in the length of the tour. Moreover the proposed BP algorithm provides superior results. The most significant improvements of solutions found by the BP is for maps *jh* and *ta*. The proposed heuristic designed with consideration of consecutive path planning leads to about twenty percents shorter tour.

The examination of the BP performance shows efficiency of the guards optimization procedure, because the procedure can be used for any guards set, solutions of other algorithms have been optimized. The overall comparison presented as ratios according to the CPP algorithm is presented in Table 4.5 and Figure 4.21. The post-processed solutions are denoted as CPP-opt and RDS-opt. The solutions of the BP algorithm without the optimization is denoted as BP-nonopt.

d	**RDS Algorithm**				**BP Algorithm**			
[m]	GR	LR	s_{GR}	s_{LR}	GR	LR	s_{GR}	s_{LR}
inf	0.38	0.76	0.05	0.05	0.38	0.77	0.05	0.05
10.0	0.49	0.85	0.08	0.07	0.45	0.74	0.11	0.13
5.0	0.61	0.89	0.13	0.11	0.54	0.75	0.12	0.14
4.0	0.65	0.89	0.15	0.11	0.60	0.76	0.15	0.14
3.0	0.67	0.91	0.09	0.05	0.62	0.78	0.11	0.13
2.0	0.75	0.93	0.04	0.03	0.59	0.82	0.04	0.04
1.5	0.72	0.88	0.03	0.04	0.56	0.75	0.02	0.02
1.0	0.70	0.85	0.01	0.04	0.60	0.74	0.01	0.02

Table 4.4: Algorithms performance, relatively to the CPP.

Algorithm	GR	LR	s_{LR}	s_{GR}
CPP	1.00	1.00	0.00	0.00
RDS	0.62	0.87	0.08	0.15
BP	0.54	0.76	0.10	0.12
CPP-opt	0.74	0.91	0.04	0.03
RDS-opt	0.53	0.84	0.08	0.12
BP-nonopt	0.64	0.77	0.11	0.18

Table 4.5: Overall performance.

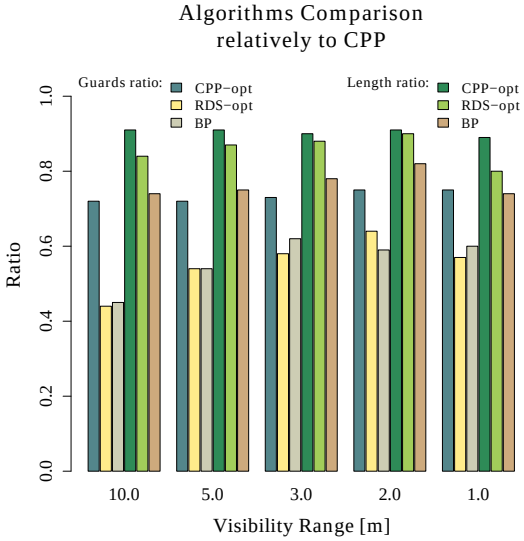


Figure 4.21: Algorithms comparison.

4.5 Discussion

Three algorithms to find a set of sensing locations with restricted visibility range have been presented in this chapter. Each particular algorithm has been evaluated and the most suitable parameters have been identified. The found parameters have been then used in the overall comparison of the algorithms. The results indicate that the new proposed algorithm called BP outperforms two other algorithms (the CPP and RDS) in quality of solution point of view. Even though the decoupled approach of the inspection planning is motivated by the high cost of sensing, the BP algorithm provides also shorter inspection paths, thus it can be eventually used for problems where the cost of sensing is not significantly dominant over the cost of motion.

The presented experimental results provide particular suitable parameters for the CPP and the RDS algorithm. Also the polygon filtering technique has been combined with the CPP to increase the solution quality. Moreover the proposed post-optimization procedure improves the solution quality found by the CPP and the RDS algorithms. The proposed

combination of the polygon filtering technique and the optimization procedure significantly improves solutions of the CPP algorithm. The CPP has the lowest computational requirements and it does not suffer by numerical stability issues, hence it is suitable for real-time applications, however the number of guards is higher.

From required computational time point of view the current implementation of the BP algorithm is very poor. It is mainly due to used geometric representation. The most time expensive part is the covering exterior in which possible best point to cover area is determined by computation of visibility polygons. The used technique can be replaced by more advanced search data structure like the shortest path map. In relation to the number of random samples of the RDS it should be remarked that the BP algorithm uses only one random sample, even though it can be easily extended to consider several samples. Despite this "handicap" the quality of solutions found by the BP outperforms the RDS even with higher number of samples.

The boundary of the BP algorithm greatly affects the algorithm performance, as it represents knowledge about the environment. The performance of the algorithm can be improved by a more sophisticated structure for the boundary. Probably more suitable structure is the Visibility-Voronoi diagram [267] or the Saturated Generalized Voronoi Diagram (SGVD) [127]. The advantage of the BP algorithm is its modularity to use any kind of boundary to improve randomization process in the first sampling stage. In this sense, it allows to use knowledge about environment in a geometrical form.

The incident constraint has not been considered. The BP algorithm can deal with additional visibility constraints, because it use similar sampling strategy like the RDS. Consideration of additional constraints is the main advantage of the sampling based algorithms.

The valuable result is a learned lesson during experimental verification of the algorithms and experimental application in the PeLoTe project. The numerical issues are critical for real applicability of the algorithm in the complex system for several robots. The real experiments also provide feedback from the real users about performance of the algorithm, a solution should be provided in less than several seconds and solution found in five seconds has been considered as slow and annoying. The inspection task, that is planning paths to search a building, is executed by the mission coordinator in a search and rescue mission to provide an allocation of particular rescuers and robots, hence it is preferred to deliver an approximate solution in required time, instead of optimal (or better) solution later.

Chapter 5

Multi-Goal Path Planning for a Group of Cooperating Robots

This chapter is dedicated to the multi-goal path planning problem that deals with finding a collision-free path to visit a set of locations in the robot workspace. Paths between goals are assumed to be known and the planning problem is considered as the Traveling Salesman Problem (TSP) and the problem of planning for a group of cooperating mobile robots as the Multiple Traveling Salesmen Problem (MTSP). In the context of the inspection task, goals can be sensing locations found by a sensor placement algorithm from the previous chapter, thus in order to accomplish the inspection task all found sensing locations have to be visited. The total mission time is important in a search and rescue scenario, thus the MinMax variant of the MTSP is preferred. To follow the TSP notation goals (sensing locations) to be visited are also called cities in this thesis.

The TSP can be formulated on a graph, in which traveling cost between two cities is considered, thus cities can be connected by an arbitrary path. The only assumption of the path is that it must be feasible for particular mobile robot. The shortest-path roadmap provides the most intuitive cost, the shortest distance between cities. Such cost can represent lower bound of time to travel, however from motion planning (or motion control point) of view, the time to travel depends on the robot kinematic and dynamic constraints. In the decoupled approach of the inspection planning, the AGP part minimizes the cost of sensing, while the TSP part minimizes the cost of motion. Moreover in the case of the MTSP, particular sensing locations are assigned to each robot in order to their cooperative behaviour. The sensing locations are found with the assumption of discrete sensing (visibility), it means the robot is stopped or its movement is very slow for precise measurement at the location. The assumptions considered in the previous chapter have to be also considered in this chapter, otherwise feasibility of the found inspection plan cannot be guaranteed. Particularly the robot has differential nonholonomic drive and can be represented by a disk in a polygonal domain, which practical means a point robot in the shrunk free space.

The chapter is organized as follows. At first, related work with focus to selected approach to solve the TSP and the MTSP with MinMax criterion are presented in the next section. The main selected method to solve the TSP is based on the Self-Organizing Map (SOM) that is extended to an environment with obstacles. Two such approaches are presented in Section 5.2 together with two additional modifications. An approach based on application of SOM on a graph is presented in Section 5.3. Two variants of the multi-goal

path planning problem in the context of the inspection task in a search and rescue mission are discussed in Section 5.5. Experimental results are presented in Section 5.6 and finally the chapter is concluded by a discussion about proposed approaches, Section 5.7.

5.1 Related Work

The TSP and MTSP are known to be NP-hard and several exact and heuristic algorithms have been proposed during several decades of studying these problems. In addition to combinatorial approaches and integer programming formulations, soft-computing techniques like neural networks, genetic algorithms, simulating annealing or ant colony optimization have been successfully applied to these problems. An overview of approaches and variants of problem formulations is already presented in Section 2.4, thus this section is focused on particular selected approaches to address the MTSP with MinMax criterion in the context of cooperative multi-goal path planning. One of them is the heuristic algorithm called GENIUS [98]. The second one is the self-organizing map (SOM) approach that has been successfully applied to the MTSP-MinMax by Somhom and et. in [243].

SOM is particularly interesting, because it does not use direct graph representation of the problem like combinatorial approaches. Weights of the neuron connections represent coordinates of nodes, that are adapted to coordinates of presented cities during learning phase. Nodes can be viewed as points moving in the environment and path is formed by connections of these nodes. The inspection task belongs to the hybrid-visibility domain, and nodes are moved inside the environment. Thus, an idea about connection between the continuous space of the inspected environment and the discretized solution of the AGP come out. Possible consideration of environment during solution of the TSP is the main reason why self-organizing neural networks have been studied and why it forms the main approach to solve the multi-goal path planning problem in this thesis.

Beside the motivation above, soft-computing techniques for the MTSP-MinMax, including SOM, have been studied in [159, 157]. Based on this preliminary comparison, SOM has been selected as the most promising approach. Although other soft-computing techniques provided better quality of solution (about units of percents), their real computational requirements were worse than for the SOM approach, despite the fact that SOM has been used in a naïve form, which is unsuitable for its application in the motivation problem.

The studied algorithms provide approximate solutions of the MTSP-MinMax. The quality of a solution can be worse due to proposed modifications of the SOM procedure that decrease the computational requirements in the related path planning problem. Thus, the exact or high quality reference solutions obtained by another approach can be beneficial to examine efficiency of the proposed modifications. The exact solution is too computationally demanding, therefore GENIUS has been chosen to provide reference solution of the MTSP-MinMax. A brief description of this algorithm is presented in Section 5.1.1.

Several ideas how to improve SOM for the TSP have been published. These ideas address issues of the SOM approach and can be particularly useful in the MTSP-MinMax, because the SOM community seems to be focused on the TSP and only the approach [243] has been found for the MTSP-MinMax variant. This fact together with the motivation of better understanding of the SOM approach are reasons why several paragraphs are dedicated to describe several SOM variants in Section 5.1.2. Besides, the description of SOM introduces reader to the terminology used.

5.1.1 GENIUS

The GENIUS algorithm is based on two heuristics: GENI (Generalized Insertion) and US (Unstringing and Stringing). It represents general heuristic and it has been used to find a solution of the MTSP-MinMax in [98]. The first heuristic is a construction method while the second heuristic is an optimization method. Tours are initially constructed by GENI. After that, the Tabu Search technique is used to exchange cities from one tour to another, while GENI is utilized for vertices insertion/removing. Finally, the US optimization procedure is applied. It removes a vertex from the tour and inserts the vertex into the same tour by the GENI algorithm. The procedure is repeated until a vertex re-insertion does not improve the quality of solution.

A parameter p of the GENI algorithm defines size of the neighbourhood that is used to select the best possible vertex insertion. Performance of the tabu search can be controlled by three additional parameters: q, Θ and T_{max}. The q parameter determines size of the global neighbourhood to select an appropriate tour for a vertex exchange, Θ controls the number of iterations for which a move of vertex according to particular tour is declared tabu. The maximal number of allowed iterations without improvement is defined by the T_{max} parameter.

p	.....	5
q	.....	5
T_{max}	.....	10

(a) GENIUS-*fast*

p	.....	14
q	.....	5
T_{max}	.....	100

(b) GENIUS-*quality*

Table 5.1: Parameters of the GENIUS heuristic algorithm for the MTSP-MinMax.

Recommended values of parameters have been suggested by authors [98] and they have been also verified in [262]. Two sets of parameters are shown in Table 5.1, the first set can be called *fast*, because it provides a good trade-off between speed of the algorithm and quality of solution. The second set provides high quality solutions, but it is computationally demanding, the set of parameters is denoted as *quality* set in this thesis or as the GENIUS-*quality* algorithm variant. For each operation stored in the tabu list the value of Θ is selected randomly from the interval $\langle 7, 27 \rangle$.

5.1.2 Self-Organizing Neural Network - SOM

The self-organizing neural network (also called Self-Organizing Map - SOM) has been select as the main technique to solve the path planning problem for a group of cooperating robots in the inspection task scenario. The main motivation of the selection has already been presented in paragraphs above, however several notes should be made in relation of the TSP and the algorithm performance. The performance of a SOM based TSP solver is poor in comparison with the classical heuristic approach, which is mention in almost every paper presenting a new SOM variant for the TSP. Authors of [56] suggested that history of heuristic approaches is more than five decades while neural network, in particular SOM, approaches have only about twenty years history and produce interesting results. Also it should be noted that one of the most powerful heuristic called Lin-Kernighan has been proposed in 1973 [182], but an efficient implementation has been proposed relatively recently by Keld Helsgaun in 2000 [118]. This observation can be motivation to not be so pessimistic to the SOM approaches, but the main motivations of SOM are its interesting

features. At first the SOM provides solution of the TSP, which is directly interpretable as a geometrical structure and the adaptation can be viewed as an exploration of the environment. This behaviour motivates to think about the solution of the Watchman Route Problem, which is also formulation of the inspection task suitable for continuous sensing model[1]. In addition, self-organizing neural networks are successfully used in various domains [216, 241], despite the fact the convergence guarantee is still under investigation, which is one of the possible issues discussed in the following sub-sections.

This section is organized as follows. The next paragraph introduces the SOM approach to the TSP and it also introduces the reader to the basic terminology. Then, an overview of SOM variants for the TSP are presented and finally SOM for the MTSP with MinMax criterion is described. The presented self-organizing schema is used in the new proposed algorithms to solve the cooperative multi-goal path planning or more specifically the cooperative inspection task problem, as a solution of the MTSP-MinMax for environments represented by a polygonal domain.

Traveling Salesman Problem

A structure of the self-organizing neural network for the TSP is a two-layered competitive learning network [242]. It contains two dimensional input vectors and an array of output units. An association between the learning network and a geometrical representation of the solution is shown in Figure 5.1. An input vector i represents coordinates (c_{i1}, c_{i2}) of the

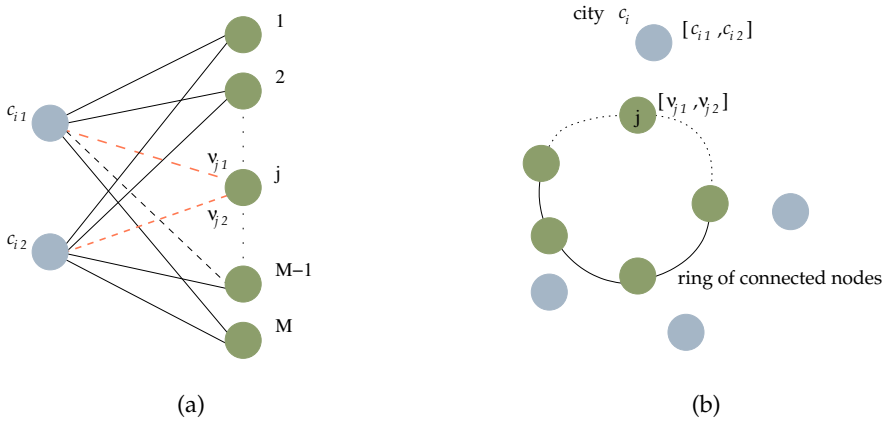


Figure 5.1: Schema of the two-layered neural network and associated geometric representation.

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. A basic schema of the self-organizing adaptation procedure for the TSP is shown in Algorithm 4.

Cities are presented to the network in a random order until the Euclidean distance of each winner node to the particular city is less than given δ. The winner node is the closest

[1] In fact, the SOM approach for the Watchman Route Problem is presented in the following chapter, but the first step has been an extension of the SOM to a polygonal domain, which is presented in this chapter.

Algorithm 4: TSP - a basic schema of the self-organizing approach

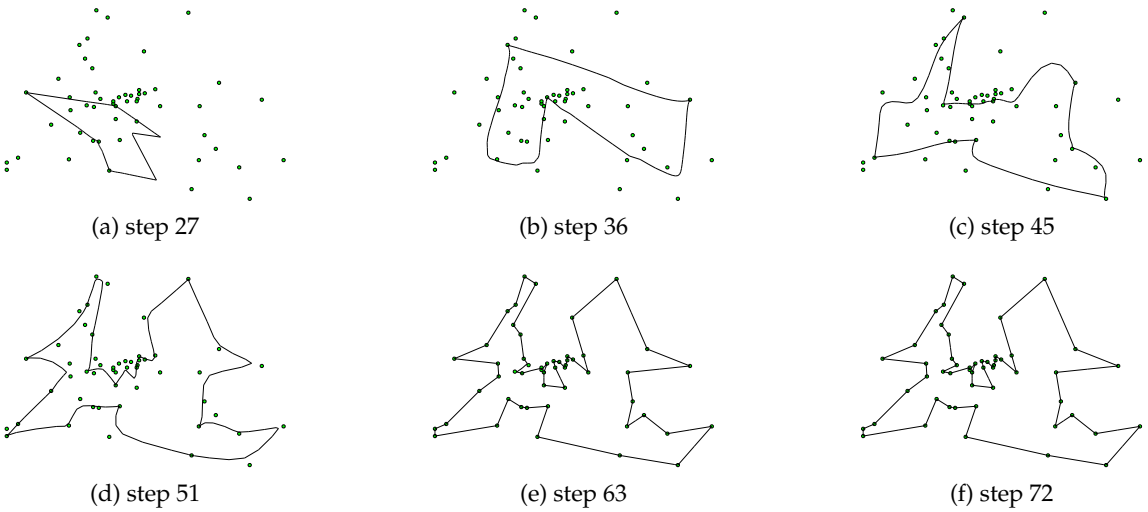
Input: $\boldsymbol{C} = \{c_1, \ldots, c_n\}$ - set of cities
Input: δ - maximal allowable error
Output: $(\nu_1, \ldots, \nu_M)$ - sequence of node weights representing city tour.

```
initialization(ν_1, ..., ν_M)                    // set weights of neurons
i ← 0                                  // reset the adaptation step counter
repeat
    error ← 0                          // clear error for new adaptation step
    Π(C) ← create a random permutation of cities
    foreach c ∈ Π(C) do
        ν* ← select winner node to the city c
        error ← max{error, |ν*, c|}
        adapt(ν*, c) // move the winner node and its neighbours towards c
    i ← i + 1                          // increment the adaptation step counter
until error ≤ δ
```

node according to distance to the city. The adaptation function `adapt` moves winner node and its neighbouring nodes towards the presenting city c_i by a equation

$$\nu'_j = \nu_j + \mu f(.)(c_i - \nu_j), \tag{5.1}$$

where μ is the fractional learning rate. The neighbouring function $f(.)$ must possess two important characteristics: it should decrease for farther neighbours and its pervasiveness should decrease during adaptation [243]. An example of the ring adaptation process is shown in Figure 5.2.



(a) step 27 (b) step 36 (c) step 45

(d) step 51 (e) step 63 (f) step 72

Figure 5.2: Performance of the SOM procedure for the TSP *berlin52* from the TSPLIB [221].

An important property of an algorithm is ability to provide a feasible solution even if the convergence condition fail. This can be guaranteed by the inhibition mechanism [242]. It was introduced to improve the convergency, but it also provides a solution of the TSP at the end of each adaptation step. During one complete presentation of all cities to the network (one adaptation step), the winner nodes are marked as inhibited and winner

nodes are selected only from non-inhibited nodes. The inhibition mark is cleared before new adaptation step. By this mechanism all cities have associated distinct winner nodes and a tour can be constructed by traversing the ring of nodes. The exact length of the tour is then found as the sum of city–city lengths.

Overview of SOM variants

One of the earliest usage of the Kohonen-type network (SOM) to solve the TSP was proposed by Angéniol et. al. in 1988 [14]. The approach follows constructive heuristic and starts with one node. A node is duplicated if it is the winner for two different cities and it is deleted if it is not selected as winner for three complete presentations of cities. Growing of the ring structure has been also used in the FLEXMAP proposed in 1991 [100], however the deletion mechanism has been omitted. The maximal number of required nodes has been close to $2.5n$, where n is the number of cities, up to problem size with 2392 cities. On the other hand, Budinich used the same number of nodes as cities and the inhibition is replaced by a real value derived from the winner node and its neighbouring nodes [39, 40]. The tour is constructed from the ordered sequence of cities according to the value.

An inhibition of nodes that are winners too often has been used in the Guilty net algorithm [44]. The inhibition mechanism was substituted by the vigilance parameters in the Vigilant Net [42] where initialization of the weights is discussed. Superior results are reported for starting positions of nodes as the convex hull approximation of the cities. Aras used geometrical properties of the ring and a topology of cities in his KNIES algorithm [19]. The algorithm uses regular adaptation of the winner node, which is moved towards the city. In addition, nodes that are not in the activation bubble (set of neighbouring nodes), which are not moved closer to the city, are moved in such a way that global statistical properties of the data points are preserved. These movements are called a dispersing phase. The proposed algorithm, called KNIES_DECOMPOSE has been used to solve large TSP instances by the decomposition of the problem into several clusters [20]. The algorithm performs in five steps. After the problem is partitioned into clusters, an order to visit the clusters is determined and entering and leaving cities for each cluster are found. An approximation of the Hamiltonian path problem in each cluster is found and finally paths are joined. The large scale problems have also been addressed in [261]. The SOM approach has been combined with the Adaptive Resonace Theory and a solution has been obtained for a problem with about fourteen thousands cities.

The presented brief overview of the SOM techniques for the TSP shows three main directions how SOM can be improved:

- efficient adaptation process,
- initialization of the weights, which should be related to the geometric structure of the Euclidean TSP,
- combination with other approaches.

Approaches concerning these directions are briefly described in the following paragraphs.

Probably the most complex SOM algorithm for the TSP is the Co-Adaptive net introduced in [56]. The algorithm uses higher number of nodes than the number of cities and it utilizes the adaptive node neighbourhood that is updated after each adaptation step. The learning process is divided into competition and co-operation phases. The co-operation phase is based on not moving of winner nodes or their neighbours more than once. The algorithm also uses the near-tour to tour construction, which creates a complete tour if the current winner nodes are not distinct. The best tour is kept during the adaptation and it is

used if the final found solution is worse. Authors presented huge set of results and comparison with other approaches and they reported that their approach outperforms other variants and together with [242] provides better results than Aras's KNIES [19], which use the statistical properties of the data points.

Even though the statistical approach (KNIES) has been outperformed, the convex hull property has been studied in the expanding SOM variant called ESOM [135, 181]. To follow the convex hull property and to design an appropriate learning rule, which will consider the global parameters of the problem, the Intergrated SOM (ISOM) uses an evolutionary principle and combines SOM with a genetic algorithm [136, 137]. The convex hull property is also studied in [277], authors used more conservative learning rule than ESOM: a movement of the nodes, which follows the expansion to preserve convex hull property, is restricted. The algorithm provides almost identical results as ESOM, but the learning rule is much simpler.

SOM has also been combined with two heuristics in [214]. The first one is called regeneration and it is a create-and-delete process. The deleted node is the *worst* node according to two parameters F and K, which are updated during adaptation. The F parameter is increased, when a winner node is already assigned to other city and decreased in other cases. The K parameter is increased by any competition involving a new target and it is decreased if a node is assigned to the city. The second heuristic is called competition strategy and it considers a segment of the ring during the computation of the nearest node to the city and creation of a new node. The proposed SOM algorithm has been use to plan trajectories for the two arms fruit-harvesting robot [215], the planning problem has been formulated as the double traveling salesmen problem.

Another studied research direction is setting of adaptation parameters: the learning rate α and the neighbouring function variance G, which is also called the gain parameter. The SETSP algorithm proposed in [260] uses different functions for α and G. The proposed algorithm is compared with the Guilty Net, Angéniol's approach and Aras's KNIES algorithms. Presented results of the proposed functions outperform all compared algorithms. Another functions have been proposed in [283] and in [26], where authors discussed influence of an initial structure of the ring [282]. Murakoshi and Sato proposed individual neighbouring functions for each node in the neighbourhood of the winner node in their multiple scale neighbourhood functions approach [199].

The SOM approach is still not competitive with classical approaches to the TSP, however many proposed approaches improve its performance. The aforementioned approaches demonstrate active research on this topic and for a comprehensive overview of previous results see [240, 56]. An overview of the recent results are well described in the new memetic neural network approach [63, 64].

To concluded overview of SOM approaches an important aspect of the related work have to be mentioned. Almost all published papers about the SOM variant for the TSP includes experimental results only for the Euclidean variant of the TSP, in which cities are placed in a plane and distances satisfy the triangle inequality. In the context of the path planning, such algorithms can be used only for environments without obstacles that means they are not practical for mobile robotics. Obstacles add difficulties to the adaptation process, because the distance between a node and presented city have to be computed with respect to obstacles, which can be computationally demanding. This issue is addressed in Section 5.2, where an exact approach based on the *shortest path map* is described, and an approximate solution of the shortest path problem in the context of the SOM adaptation procedure is introduced. Only one approach that deals with obstacles

has been found in literature: an adaptation rule for the TSP problem on a graph is presented in [274]. The approach is discussed in Section 5.3.

Multiple Traveling Salesmen Problem

An individual ring of nodes is created for each salesman in the MTSP. All tours start and end at the same city called *depot*. The adaptation process must ensure that all tours are connected with the depot, therefore an individual adaptation of the winner node from each ring to the depot is necessary. A schema of the adaptation procedure with the inhibition mechanism is depicted in Algorithm 5.

Algorithm 5: MTSP - self-organizing approach

Input: $\boldsymbol{C} = \{c_d, c_2, c_3, \ldots, c_n\}$ - set of cities, where c_d is the depot
Input: m - number of salesmen
Input: (d, G, μ, α) - parameters of the self-organizing neural network
Input: δ - maximal allowable error

$\boldsymbol{R} \leftarrow \{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_m | \boldsymbol{r}_i = \text{ring}(c_d)\}$ `// initialization of rings of nodes`
$i \leftarrow 0$ `// reset the adaptation step counter`
repeat
  $error \leftarrow 0$ `// reset error for the current adaptation step`
  $I \leftarrow \emptyset$ `// set of inhibited nodes`
  foreach $\boldsymbol{r} \in \boldsymbol{R}$ **do**
    $\nu^\star \leftarrow$ *select winner node from the ring* $\boldsymbol{r}$ *to the city* c_d, $\nu^\star \notin I$
    $error \leftarrow \max\{error, |\nu^\star, c_d|\}$
    $\text{adapt}(\nu^\star, c_d)$ `// move` $\nu^\star$ `and its neighbours towards` c_d
    $I \leftarrow I \cup \{\nu^\star\}$ `// inhibit the winner node`
  foreach $c \in \Pi(\boldsymbol{C} \setminus \{c_d\})$ **do** `//` $\Pi(\boldsymbol{C})$ `is a random permutation of cities`
    $\nu^\star \leftarrow$ *select winner node to the city* c, $\nu^\star \notin I$
    $error \leftarrow \max\{error, |\nu^\star, c|\}$
    $\text{adapt}(\nu^\star, c)$ `// move` $\nu^\star$ `and its neighbours towards` c
    $I \leftarrow I \cup \{\nu^\star\}$ `// inhibit the winner node`
  $G \leftarrow (1 - \alpha)G$ `// decrease the gain parameter`
  $i \leftarrow i + 1$ `// increment the adaptation step counter`
until $error \leq \delta$

Authors of [243] proposed a competitive rule for the MinMax criterion. The rule prefers nodes from shorter rings:

$$\nu^\star = \text{argmin}_\nu \, |c, \nu| \cdot \left(1 + \frac{dist_\nu - avg}{avg}\right), \tag{5.2}$$

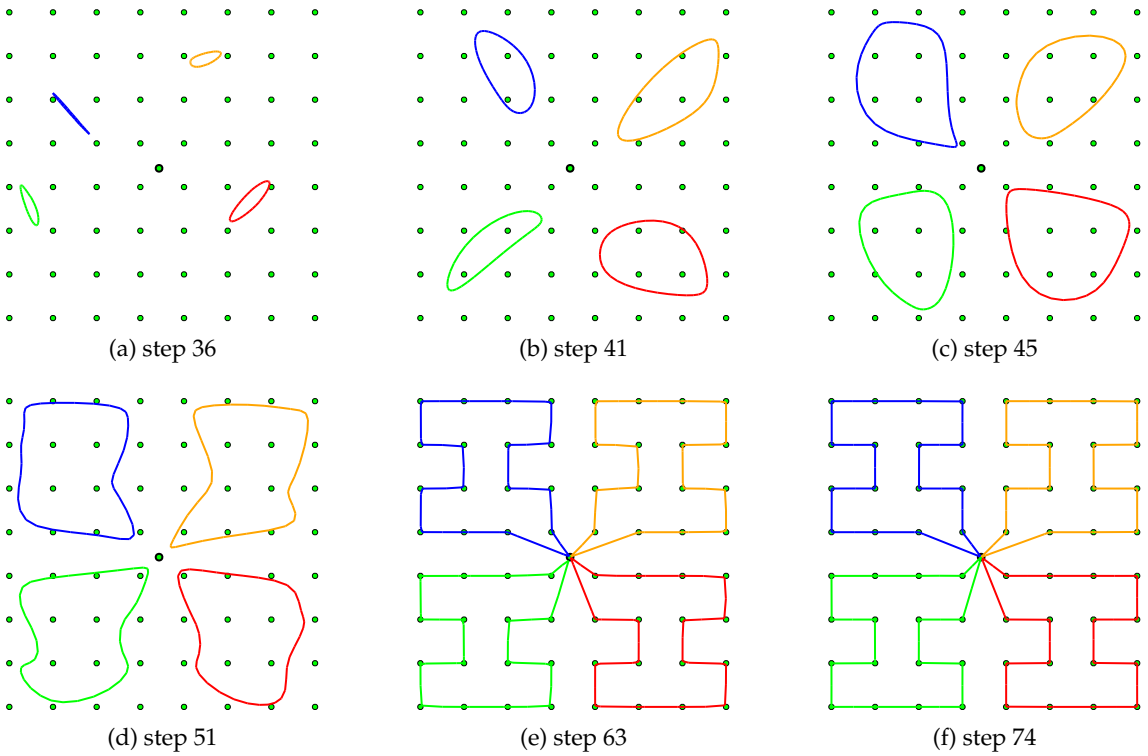
where $|.|$ denotes the Euclidean distance between the city c and the node ν, $dist_\nu$ is the length of the ring into which the node ν belongs, and avg is the average length of the rings. The authors also recommend following neighbouring function

$$f(G, d) = \begin{cases} e^{-\frac{d^2}{G^2}} & d < 0.2M, \\ 0 & \textit{otherwise}, \end{cases} \tag{5.3}$$

where G is the gain parameter, d is the cardinal distance measured along the ring and M is the number of nodes in each ring. The gain G is decreased after each complete presentation of the cities to the network according to the gain decreasing rate α. An appropriate

initial value of G depends on the problem size and authors experimentally found the linear relation $G_0 = 0.06 + 12.41n$. Recommend values of learning and decreasing rates are $\mu = 0.6$ and $\alpha = 0.1$.

The algorithm performance for the Euclidean MTSP-MinMax is shown in Figure 5.3.



(a) step 36 (b) step 41 (c) step 45

(d) step 51 (e) step 63 (f) step 74

Figure 5.3: Performance of the SOM algorithm for the MTSP-MinMax without obstacles.

The presented self-organizing procedure together with recommended values of its parameters is the main SOM schema used in this thesis for the multi-goal path planning for a group of cooperating robots. The particular parameters are summarized in Table 5.2. The required cooperation is represented by the MinMax criterion, which leads to distribute movements among robots in order to minimize the total mission time. The selected approach is able to provide a solution for problems with hundreds of cities, which is sufficient for the inspection task (according to the results presented in the previous chapter), however other SOM approaches have been applied for larger problems.

Parameter			Description
M	=	$\frac{2.5 \cdot n}{m}$	the number of neurons in the each ring
G_0	=	$12.41 \cdot n + 0.06$	the initial value of the gain parameter
d	=	$0.2 \cdot M$	the parameter of the neighbouring function
μ	=	0.6	the learning rate
α	=	0.1	the decreasing rate
δ	=	0.001	the minimal required distance of the winner from the city

Table 5.2: Parameters of the SOM algorithm for the MTSP, where n is the number of cities and m is the number of salesmen.

Convergence of SOM

The following paragraphs are dedicated to discussion about convergence of the SOM approach. Mainly to note related aspects and also to provide a different point of view to the SOM adaptation instead of the geometrical interpretation, which is the motivation of the proposed SOM based algorithms in this thesis.

In [60], authors presented review of the convergence results. A generalization of the SOM competitive learning can be considered as a vectorial quantization algorithm. Authors discussed measures of the self-organization and for the discrete case they cited Ritter's work on potential $V_n(m)$ as the true measure of the self-organization. The algorithm can be viewed as an approximation of the stochastic gradient algorithm derived from the function $V_n(m)$. It is noted that many applications of the SOM algorithm are based on non-singular gradient of $V_n(m)$, which correspond to the Kohonen algorithm, and a singular part, which prevents the algorithm from being gradient descent.

The most studied convergence properties are for the one dimensional case. The convergence proof is based on the Markov chain technique. Sadeghi presented study of multidimensional Kohonen algorithm in [223]. He used the fact that in each step of the learning phase the only needed information are those of the last step, which is the Markov process consideration. A constant learning rate is assumed to use results from the homogeneous Markov chains. It is shown that Deoblins's condition is valid for the multidimensional Kohonen algorithm considered as Markov process evolved in a topological space.

More recent overview of neural networks properties can be found in [116] where convergence results and the topology preservation are discussed. Mathematical exact definition of the topology preservation (topographic) mapping from an input space V to an index set A of reference vectors W is given by the same authors. They noted that topographic product and its derivatives seem to be the best tools for practical use, although they are not based on the mathematical exact definition.

SOM in the Multi-Goal Path Planning

However the convergence property is desired, with respect to applications of the mobile robotics an approximate solution may be sufficient, e.g. an approximation that is found after finite number of adaptation steps. Mainly due to the following aspects. The planned path is found in a model of the environment, which can be almost every time somehow modified or it can contain dynamic objects, that are not considered during planning. Paths can be replanned according to new information about the environment. In real application, it is also required to have a solution at requested time, rather than to have better or optimal solution later. From this perspective, the planned paths can be viewed as guidelines, in which directions each robot should travel instead of the exact "optimal" plan. To avoid the critique be too skeptical, it is clear that more accurate approximation provides more valuable solution.

5.2 Solving MTSP by SOM in a Polygonal Domain

The multi-goal path planning is the second part of the decoupled approach of the inspection task. Cities in the MTSP are sensing locations found in a polygonal domain, thus paths between cities must by traversable by a mobile robot and have to navigate the robot among obstacles. The shortest path among obstacles can be called Euclidean geodesic [48] or the *geodesic path* [145]. Similarly to the sensor placement, a polygonal map may be expanded by a disk representing the robot size. All cities have to be inside the free space reachable by the robot and without lost of generality only one connected component can be assumed[2]. The polygon filter technique based on relevance measure, Section 4.2.2, may be used to remove unnecessary vertices of polygonal representation of the environment. Fast computation of the shortest path is crucial for a reasonable required computational time of the SOM adaptation. The path (distance) queries are used in two situations.

1. Determination of a (shortest) distance (path) between a node and the city is a part of the winner selection process. The path is also utilized while a node is moved towards the city in the adaptation part, i.e. new node' position (weights) is on the path closer to the city according to the neighbouring function.
2. Determination of a (shortest) path between two nodes is used in the computation of the length of the ring. The length is used as a weight of the node–city distance in the competitive rule for the MTSP-MinMax.

The naïve approach of finding the shortest node–city path may be based on computation of the visibility graph that can be found in $O(n \log n)$, where n is the number of polygon vertices. The shortest path can be then found in the graph by Dijkstra's algorithm in $O(|e| + n \log n)$, where e is the number of edges of the full visibility graph.

The computation of the visibility graph and the shortest path can be supported by an underlying triangulation of the polygonal environment. Then, the shortest path can be found in $O(h^2 \log n + T)$, where h is the number of obstacles, n is the number of vertices and T is the time to find the supporting triangulation [222]. The shortest path can also be found by the wavefront propagation based on the continuous Dijkstra paradigm [194]. The propagation can be used to create supporting structure for the shortest path queries with respect to city, the structure is called *Shortest Path Map* (SPM). The SPM in the SOM based MTSP algorithm has been introduced in [161].

The SPM supports point–city queries, but it cannot be directly used for node–node queries, that is why other approaches have been investigated. Another reason is related to the numerical stability and complexity of several SPMs in the case of large TSP. Issues of the SPM approach are discussed in the next section and an approach based on approximation of the shortest path is presented in Section 5.2.2. Determination of length of the ring is discussed in Section 5.2.4 and an alternative stop condition of the adaptation procedure is presented in Section 5.2.5.

5.2.1 Shortest Path Maps Approach

The SPM is a planar division of the free space with respect to a point, which allows determination of the length of the shortest path to the point in time $O(\log n)$ and the shortest

[2]The case with several connected components can occur if size of the robot is large and a sensing location is placed inside an unreachable part of the environment, after obstacles expansion by the radius of the robot.

path in time $O(\log n + k)$, where n is the number of vertices representing polygonal obstacles and k is the number of bends in the path [195]. An example of the SPM is shown in Figure 5.4. The SPM supports node–city query and for a given polygonal domain $\mathcal{W}$

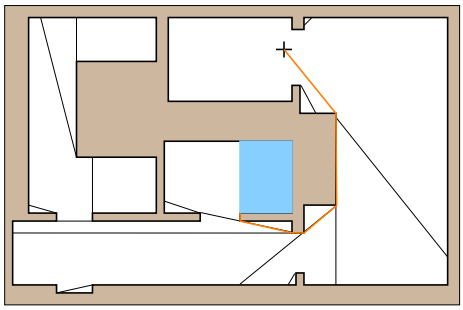
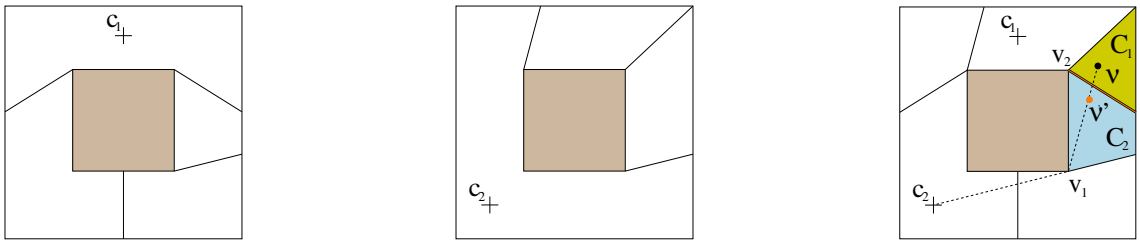


Figure 5.4: An example of the SPM, the blue rectangle represents one of the cell of the SPM division, and shortest paths from the cell to the point (cross) follow the orange path.

with v vertices, n cities, N neurons and d neighbouring neurons it can be used as follows. For each city c_i the $SPM(c_i)$ can be found in $O(nv \log v)$ [119]. To determine a distance between a node ν and c_i a particular cell of the $SPM(c_i)$ have to be found for ν in $O(\log v)$ by a point-location algorithm or in the average complexity $O(1)$ by the Bucket method [80]. A node ν is moved towards the city c_i along the shortest path in $O(k)$, where k is the number of path vertices, the cell (path) is already found in the distance query. The complexity of one adaptation step depends on a winner selection and movements of the winner and its neighbouring nodes towards the city. The winner node is found in $O(Nn \log v)$ and movements of nodes can be bounded by $O(ndv)$. The number of neurons is derived from the number of cities, e.g. $N = 2.5n$, thus the overall complexity of one adaptation step can be bounded by $O(n^2 \log v + ndv)$.

An overlay of SPM divisions (for all cities) can be used to avoid necessity of determination of the node cell for each city presentation. If a node is moved in such polygonal division, the cell can be determined during the node movement along the shortest path to the city, by similar procedure to the visibility walk in a triangulation [72]. The principle is illustrated in Figure 5.5, where an overlay of two SPMs for cities c_1 and c_2 are shown. A



(a) SPM for the city c_1 (b) SPM for the city c_2 (c) overlay of SPMs

Figure 5.5: A neuron ν adaptation to the city c_2 and its movement in polygonal division.

node ν is in the cell C_1 and it is moved towards c_2 over polygon vertex v_1. The node passes edge of the cell during its movement, therefore a cell C_2 may be determined. The cell C_2 has associated the shortest path to c_2 over v_1 and to city c_1 over v_2. A location of ν' after its movement can be considered to express complexity of the adaptation step. At first, vertices between which ν' is placed are determined in $O(v)$, then a cell is determined in $O(p_c)$, where p_c is the number of cells between two vertices. The complexity of the node moving

process can be bounded by $O(v + p_c)$ or by $O(\log v + \log p_c)$ if distances between path vertices and edges of the passed cells are precomputed. The complexity of the adaptation step for an overlay of SPMs can be bounded by $O(nd(v + p_c))$ or $O(nd(\log v + \log p_c))$.

One division seems to be more suitable for the node–city queries than an individual SPMs, however it cannot be simple stated that one overall division is better, because robustness should be also taken into account. A polygonal division found as the SPM for a polygonal map of the environment with presence of obstacles contains so-called Steiner points. The number of these new points depends on the number of reflex vertices of the map and it dramatically increases after two wavelets are merged [195]. The number of such points is even greater for an overlay of SPMs. From the numerical issues point of view, such a division contains many very narrow neighbouring cells, see Figure 5.6 for an example. An additional issue of the SPM approach relates with the required memory to

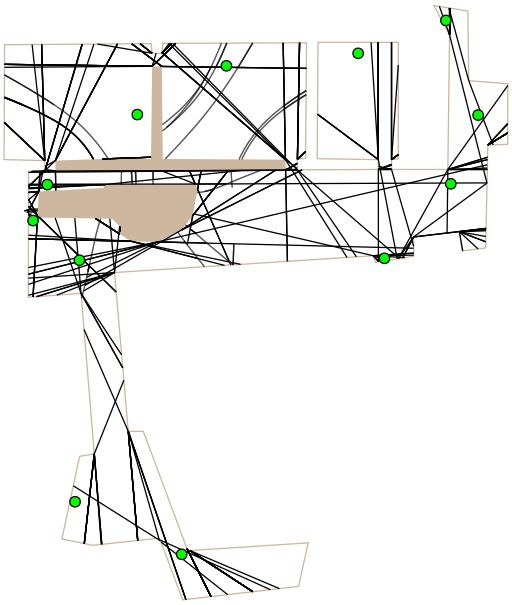


Figure 5.6: An example of SPM division overlay.

store the supporting structures. The memory requirements increases with number of cities and with the number of map vertices. Regarding the results in Chapter 4, the number of cities in the inspection task can be several hundreds for typical indoor environments and small visibility ranges. The same number of SPMs have to be created and after that, an overlay can be eventually computed.

Despite the mentioned issues the individual SPMs approach has been used in a search and rescue scenario [160, 161] for problems with less than one hundred cities.

5.2.2 Approximate Solution of the Shortest Path Query

An evolution of the ring during the adaptation process can be viewed as an exploration of the problem topology, see Figure 5.3. In early adaptation steps, nodes travel across the problem domain, while in final stages, due to lower value of the gain G, weights (nodes) changes are very small. This observation leads to expect that a rough approximation of the shortest path in early stages should be sufficient, because position of a node is dramatically changed during its movement. An approximation can also be useful to support

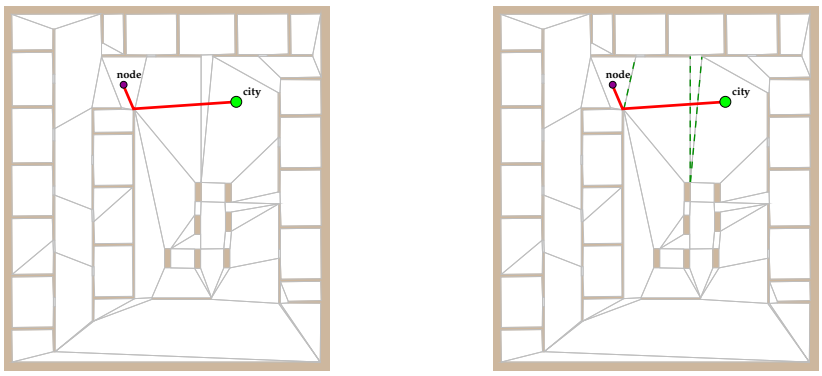
the node–node path queries required in the MTSP-MinMax variant to determine lengths of the rings. On the other hand, winner nodes are very close to cities in final adaptation steps. The motivation to use an approximation of the shortest path is a reduction of the required computational time of a TSP solution, and of course elimination of necessity of constructing complex supporting structures like the SPM. The application of the shortest path approximation is not new and it is investigated in the finding approximate geodesic path for complex 3-d shapes represented as triangular meshes [247, 253].

An approximation of the shortest path can be based on a convex polygon partition of the free space. A convex partition can be found in $O(v \log v)$, where v is the number of polygon vertices [233]. An advantage of the convex cells is that the shortest distance between a node inside a cell and a vertex of the cell is the Euclidean distance. A node is always placed in the free space, therefore it is always placed in some cell of the partition. An approximation of the shortest path between a node and the city can be found as the shortest path over a vertex of the cell.

More formally, assume a polygonal workspace $\mathcal{W}$ with v vertices and let $\mathbf{P}$ is a convex polygon partition of $\mathcal{W}$ into convex cells, $\mathbf{P} = \{C_1, C_2, \ldots, C_k\}$, where each cell C_i is represented as a sequence of polygon vertices, and a node ν is in the cell C_ν. The 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

$$w = \text{argmin}_{w_i \in C_\nu} |\nu, w_i| + |S(w_i, c)|, \tag{5.4}$$

where $|.|$ denotes the Euclidean distance between two points and the length of the shortest path between two vertices, resp. vertex and city. An example of the approximate shortest path is depicted in Figure 5.7.



(a) (b)

Figure 5.7: An approximation of the shortest path and incident diagonals.

All shortest paths from vertices to cities can be found in the visibility graph by Dijkstra's algorithm in $O(n(v+n)\log(v+n))$, where n is the number of cities and v is the number of vertices. The graph can be found in $O((v+n)^2)$ by the algorithm [210]. A point-location algorithm can be used to find C_ν in $O(\log v)$. Similarly to the SPM approach the cell after node movement can be determined by the same walking technique, but the number of passed cells is expected to be lower. A node always pass diagonals during the movement, because the node is moved from a vertex to the city along edges of the visibility graph, thus the diagonals passed by the shortest path can be precomputed. 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, i.e. the number of

passed diagonals. The number of cells does not depend on cities, because convex polygon partition is a property of $\mathcal{W}$.

A precision of the proposed approximation depends on the size of convex polygons and can be insufficient for an arbitrary point. One of the key aspect of the proposed approximation is the adaptation process. Nodes are attracted to cities during the adaptation and the winners are almost at the same places as the cities in the final adaptation steps. In these situations, the city is typically directly visible from the winner node and the Euclidean distance can be used. A refinement of the path based on this observation is described in the next section.

5.2.3 Refinement of the Approximation of the Shortest Path

The proposed approximation of the shortest path can be extended to provide shorter path or in several cases the exact solution. The shortest path is provided if a city and a node are in the same cell. Also the solution is exact if the node is at the shortest path from a vertex to the city. In other cases, an additional path refinement may be based on a test of direct visibility between the node and a vertex of the primal (rough) path.

Assume a node ν inside the cell C_ν and an approximation of the shortest 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. The refinement procedure is depicted in Algorithm 6. The visibility test is similar to the method described in [143], which is based

Algorithm 6: Refinement of the Approximate Shortest Path

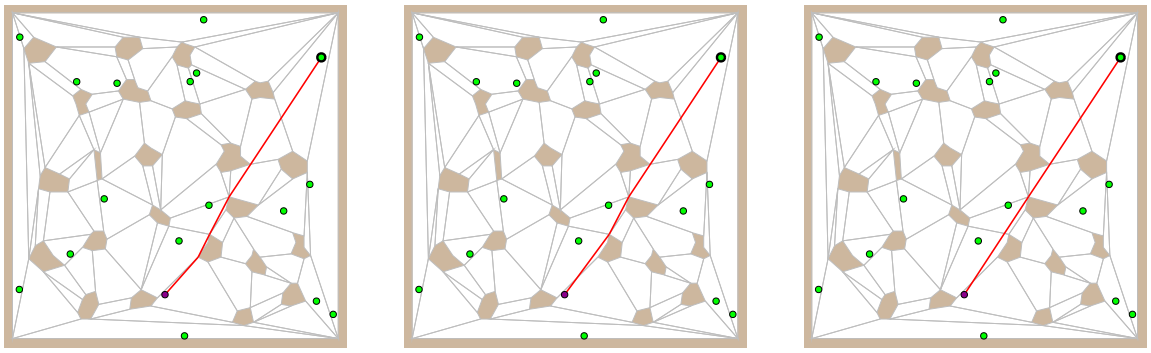
Input: ν - node, which is inside cell C_ν
Input: $(v_0, v_1, \ldots, v_k)$ - an approximation of the shortest path from ν to v_k, $v_0 \in C_\nu$
Output: $(v_i, v_{i+1} \ldots, v_k)$ - a refined path, $i \geq 0$

$i \leftarrow 0$
while visible$(\nu, v_{i+1}) \wedge i < k$ **do**
  $i \leftarrow i + 1$

on the straight walk in a triangulation [72], instead of triangulation the convex partition is used. If a straight line from ν to the vertex v_k cross 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 refined path are shown in Figure 5.8, the differences between paths are less than 0.3%.

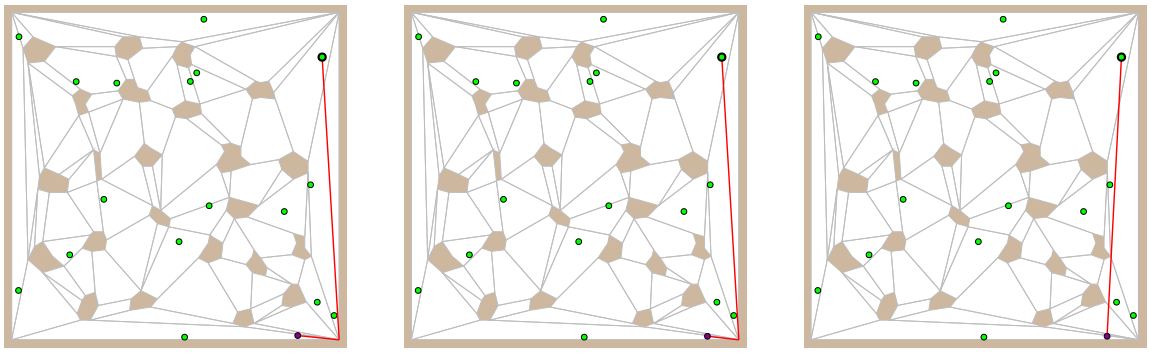
The complexity of the path refinement depends in the worst case on the number of vertices and it can be even worse than a construction of the SPM, however the real required computational time is expected to be lower. For example nodes are very close to cities in final adaptation steps, hence the node is in the same cell as the city or it is just in the next cell and only one direct visibility test should be sufficient. Also if a node and the city is not directly visible, after the node is moved towards the city along the found path, the city becomes visible and the path refinement provides the shortest path, see Figure 5.9.

The discussed examples are motivation to study the proposed approximation and path refinement in the context of the inspection task, where cities are not placed arbitrarily, they are not typically placed in the worst configuration. A set of cities is found as a solution of the AGP and one can expect that cities are placed in such a way that each city guards certain portions of the free space. The error of the approximation can be expected in the



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

Figure 5.8: Example of path refinement, gray segment represents diagonal of the convex partition, green points are cities, purple point is a node, red segments represents approximation of the shortest path between the node and the city.



(a) (b) (c)

Figure 5.9: Example of a path approximation and its improvement during adaptation: (**a**) node and city are not directly visible, (**b**) after movement of the node towards the city, (**c**) the city is directly visible and path refinement decreased the length about 10%.

first adaptation steps, but in this adaptation phase the direction of the node movement is more important than the exact shortest path. In the final steps of the adaptation, the approximation should be sufficient, because the winner nodes are very close to the cities. The expected behaviour of the proposed approximation is experimentally verified for a set of problems in Section 5.6.

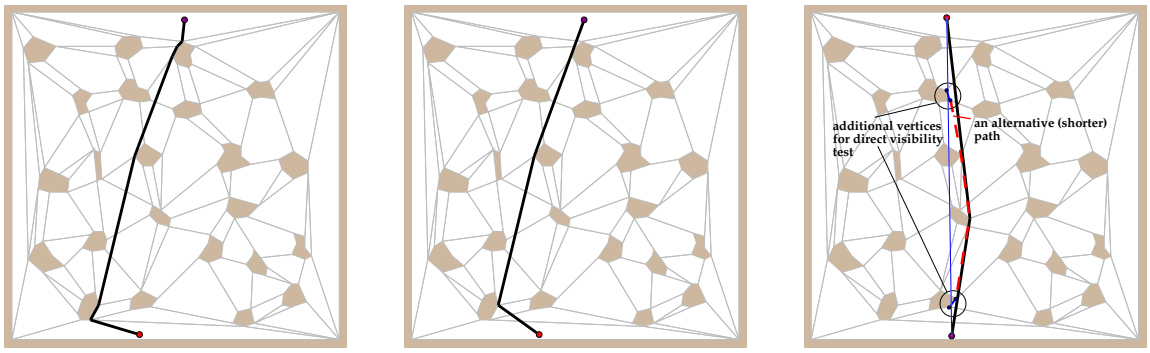
5.2.4 Determination of the Ring Length

Determination of the ring length is necessary in the MTSP-MinMax to prefer selection of nodes from shorter rings, therefore a node–node distance has to be computed. Similarly to the previous approximation of node–city path, the convex partition can be used to find an approximation of the shortest path between two nodes (points).

Assume a node ν_1 in the cell C_1 and a node ν_2 in the cell C_2. A path between ν_1 and ν_2 may be constructed from the shortest path between vertices of each cells, $S(w_1, w_2)$, where $w_1 \in C_1$ and $w_2 \in C_2$. Vertices w_1, w_2 are selected according to minimization of the length of the path

$$|\nu_1, w_1| + |S(w_1, w_2)| + |w_2, \nu_2|.$$

Such a path can be refined in similar manner like a node–city path, an example is shown in Figure 5.10. The approximation can be improved if vertices of edges that intersect the segment (ν_1, ν_2) are considered in the refinement procedure. The edges can be found by the walking procedure used in the direct visibility test for both directions: $\nu_1 \rightarrow \nu_2$ and $\nu_2 \rightarrow \nu_1$. If the incident vertices of the edges are directly visible from ν_1 or ν_2, then an alternative path over these vertices can be constructed, see Figure 5.10c.



(a) rough path (b) refined path (c) incident vertices

Figure 5.10: Examples of the node–node paths.

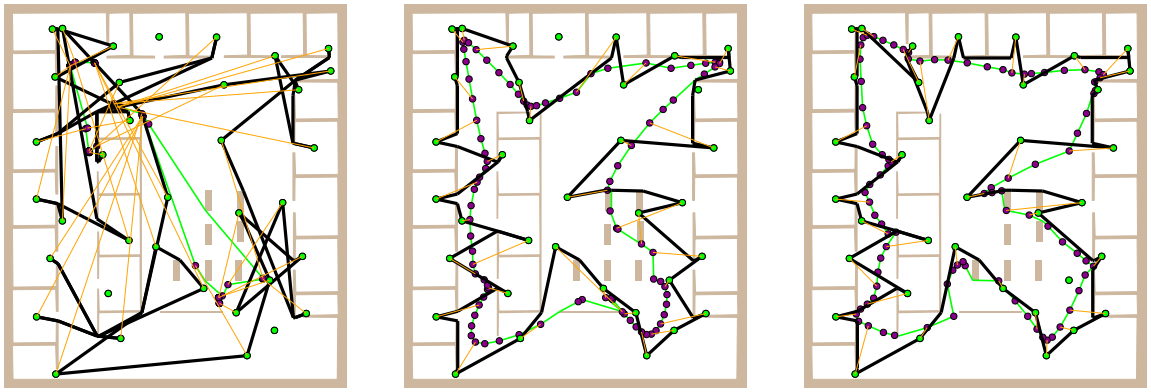
The complexity of the proposed approximation can be worse than the complexity of the SPM construction, because direct visibility tests are performed independently. However lower real computational requirements are expected, mainly due to the fact that the approximation is used only for distances between two neighbouring nodes, which are typically very close, thus they are mostly in the same cell or in the neighbouring cell.

The advantage of the proposed node–node path approximation is its dependence only on the convex partition and the visibility graph of the environment. The cities are not used, therefore supporting structures depends only on $\mathcal{W}$. If the number of cities is very high, e.g. several thousands, the proposed node–node path can also be used for the node–city paths to avoid storing the precomputed paths from vertices to cities.

City Tour Represented by the Ring

The length of the ring can be estimated in less computational intensive way in comparison to the approximation of the node–node path described above. The ring represents a tour over cities and length of the tour can be used as the weight in the competitive rule of the MTSP-MinMax variant. The shortest paths between all cities are precomputed and a winner node to each city is maintained in order to determine the length of tour represented by the current ring in $O(M)$, where M is the number of nodes in the ring.

The winner is associated to the city until a new winner is selected. A tour represented by the ring is constructed from the associated nodes along the ring. If a node has been selected as a winner in the previous adaptation step and to another city in the current step the association with the previous city is cleared to reflect change of the ring shape. Particular winners can be pulled away from the cities during movements of another winner, because they can be in its neighbourhood. So, the tour is only an approximation of the current ring. In final steps of the adaptation, most of the winners are preserved over the adaptation steps and the approximation is more accurate.



(a) step 12 (b) step 42 (c) step 46

Figure 5.11: Examples of tours represented by the ring, the ring is in green and tour represented by the ring is composed from bold black line segments.

Examples of associated tours are shown in Figure 5.11, where only one salesman is considered to illustrate the tour association. The first figure shows tour after 11 complete presentation of all cities during the 12^{th} presentation. However the ring does not cross, tour has several crossings and does not visit all cities. After several adaptation steps the tour is complete and finally it is same as the ring.

5.2.5 Alternative Stop Condition of the SOM Adaptation Process

Algorithm 5 is terminated if all winner nodes are sufficiently closed to the particular city. Additional stop conditions have been proposed in [56], because in cases of large *error*, changes of nodes position can be very small due small value of the gain G. Authors stopped the adaptation if G is equal or less than 0.01 or if nodes are in the same position they were at the end of the previous adaptation step.

An observation of the ring evolution is inspiring to propose a more simplified form of the stop condition. In the last steps, *error* and G can be still sufficiently high to continue with the adaptation process, however the winner nodes association to cities can be preserved. This observation motivates to omit the inhibition mechanism and stop the adaptation if all cities have the unique winner node in the single complete presentation of cities. The inhibition mechanism cannot be used, otherwise the adaptation is stopped after the first step. Therefore a node can be winner more than once in one adaptation step.

In relation to the propose modifications, it is expected that the convergence of SOM is not guaranteed and the stop condition based on the maximal number of adaptation steps is necessary. The final solution is obtained as tour over cities along the ring, which are associated to particular winner nodes. If the algorithm is terminated before unequivocal tour can be obtained, the complete tour is found by the following procedure.

1. Associate current winner nodes to particular cities.
2. Find the closest (not already associated) node to each city without node.
3. The tour is determined as a sequence of associated cities to nodes along the ring.

The procedure is not necessary if the inhibition mechanism is used, because at the end of each adaptation step distinct nodes are associated to cities.

The SOM adaptation procedure with the proposed stop condition is used like a constructive heuristic to obtain an initial feasible solution of the TSP, because even if a node is winner, its position can be changed by the adaptation of neighbouring nodes during same adaptation step. The inhibition has been introduced to avoid nodes to win too often and from this point of view the proposed stop condition can be like a step back. Detail comparison of this approach has not been found in literature. That is why this approach, called *unique*, is experimentally verified in the context of the inspection task.

5.3 SOM Adaptation Rule for a Graph Input

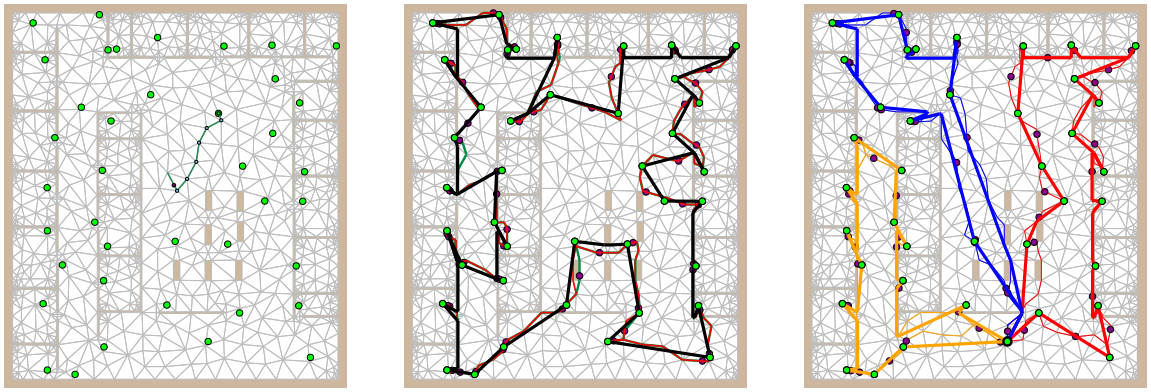
A self-organizing map for a graph input has been presented in [274] and the proposed adaptation rule is demonstrated in the TSP with obstacles. The adaptation process can be described as movements of nodes along edges of the graph, a node always lies on a graph edge (between two graph vertices). Cities are represented by particular graph vertices and a winner node is selected according to the shortest path in the graph. The graph is constructed by the topology representing network [187], which uses the Hebbian adaptation with the winner-take-all competition rule and provides induced Delaunay triangulation. However a solution of the TSP with 50 cities is presented in the paper, the paper lacks serious performance comparison. That is the main reason why following algorithm is proposed and experimentally verified.

For a set of cities C inside polygonal representation of the robot workspace $\mathcal{W}$ a graph $G(V, E)$ is constructed in two steps.

1. A triangular mesh of $\mathcal{W}$ is used to create an initial graph $G(V, E)$.
2. A triangle is found for each city $c \in C$, the city c becomes a vertex of the graph $G(V, E)$ and edges connecting c with a particular vertex of the triangle are added to the set of edges E.

A node ν lies on an edge $e \in E$ with incident vertices v_i and v_j during the adaptation. The shortest path from ν to a city c is determined as the shortest path over one of the incident vertices according to distances $|(\nu, v_i)| + |(v_i, c)|$ and $|(\nu, v_j) + |(v_j, c)|$. The winner node (and its neighbouring nodes) are moved towards the presented city along the edges of the graph. A path from a vertex v to the city c can be found by Dijkstra's algorithm in $O(N_T \log N_V)$, where N_T is the number of triangles and N_V is the number of graph vertices. The adaptation of ν towards c can be done in $O(k)$, where k is the number of paths vertices. An example of the shortest path from a node to city is shown in Figure 5.12a. A schema of the adaptation procedure is identical to Algorithm 5, the only change is the computation of the shortest path. The ring of nodes representing a solution of the TSP can be used to obtained a sequence of cities, and the final tour can be then constructed from the visibility graph. An example of a TSP solution is shown in Figure 5.12b. Notice differences between a connected ring of nodes (in green) and connected cities as a sequence of graph edges (in red), the final solution from the visibility graph is shown in black. An example of a MTSP solution is shown in Figure 5.12c.

As can be seen from Figure 5.12 lots of triangles are not necessary, because their edges are never used for the adaptation. The used triangular mesh is constructed with desired minimal angle, and these triangles are created due to small obstacle edges. A polygon filtering technique presented in the previous chapter can be used to decreased the number of triangles. Less number of triangles can decrease the computational time, but the improvement is negligible for small problems.



(a) the shortest path in the mesh (b) TSP (c) MTSP

Figure 5.12: SOM solutions of the TSP and the MTSP on the triangular mesh.

The complexity of the winner selection can be decreased if all shortest paths between each vertex and all cities are precomputed and associated to each vertex of the graph. The shortest path from ν to c can be then determined in the constant time. In addition, the adaptation of ν towards c can be performed in $O(\log k)$ using the binary search algorithm [59]. Although these optimizations may reduced the computational burden, they also increase the required memory and they are unsuitable for large graphs with thousands of vertices.

5.4 Quality of Cooperative Plan

The MinMax variant of the MTSP leads to minimize the longest tour and one can expect that other tours will be shorter or equally long. If lengths of tours are equal then all robots have to travel same distance to accomplish the task. To measure quality of cooperation a percentage deviation of lengths of tours can be used to provide a scale independent metric for particular solution of the MTSP. The measure can be called the *Cooperative Quotient* (CQ), and its zero value means "ideal cooperation". The *Collaborative Effort* (CE) may be introduced to consider the total traveled distance by all robots. A solution for m robots (salesmen) consists of m tours with lengths $l_1, \ldots, l_m$ and quality metrics are defined as follows:

- *Maximal length*:
$$L = \max\{l_1, \ldots, l_m\}.$$
- *Cooperative Quotient*:
$$CQ = \frac{s_L}{\overline{L}},$$
where s_L is the root of the sample variance, $s_L^2 = \frac{1}{m-1}\sum_{i=1}^{m}(L_i - \overline{L})^2$ and $\overline{L}$ is the average value of the lengths, $\overline{L} = \frac{1}{m}\sum_{i=1}^{m} L_i$.
- *Collaborative Effort*:
$$CE = \sum_{i=1}^{m} l_i.$$

A cooperation can be defined as a planning to maximize the utility, while the coordination is a planning to share a common resource. The problem formulated as the MTSP

represents the cooperation part of the multi-goal path planning. The common resource is a free space, where robots perform their movements. A plan that satisfy the coordination constraint needs consideration of robot position in time, hence detailed model of robot motion is necessary. The coordination problem is not explicitly addressed by the proposed formulation of the multi-goal path planning, it can be solved independently after paths are found [52], e.g. by a prioritized planning [176]. To guarantee the coordination the planned paths have to be cross-free, except the common depot, which is assumed to be approximation of initial positions of robots.

5.5 Multi-Goal Path Planning Problem Variants

This section presents two variants of the path planning problem, which are particular modification of the multi-goal path planning problem formulated as the MTSP. The problems are motivated by practical needs of the path planning in a search and rescue mission. Both problem variants are solved by the SOM algorithm for the MTSP-MinMax and demonstrate flexibility of the SOM approach.

5.5.1 Path Replanning - Multi-Depot MTSP

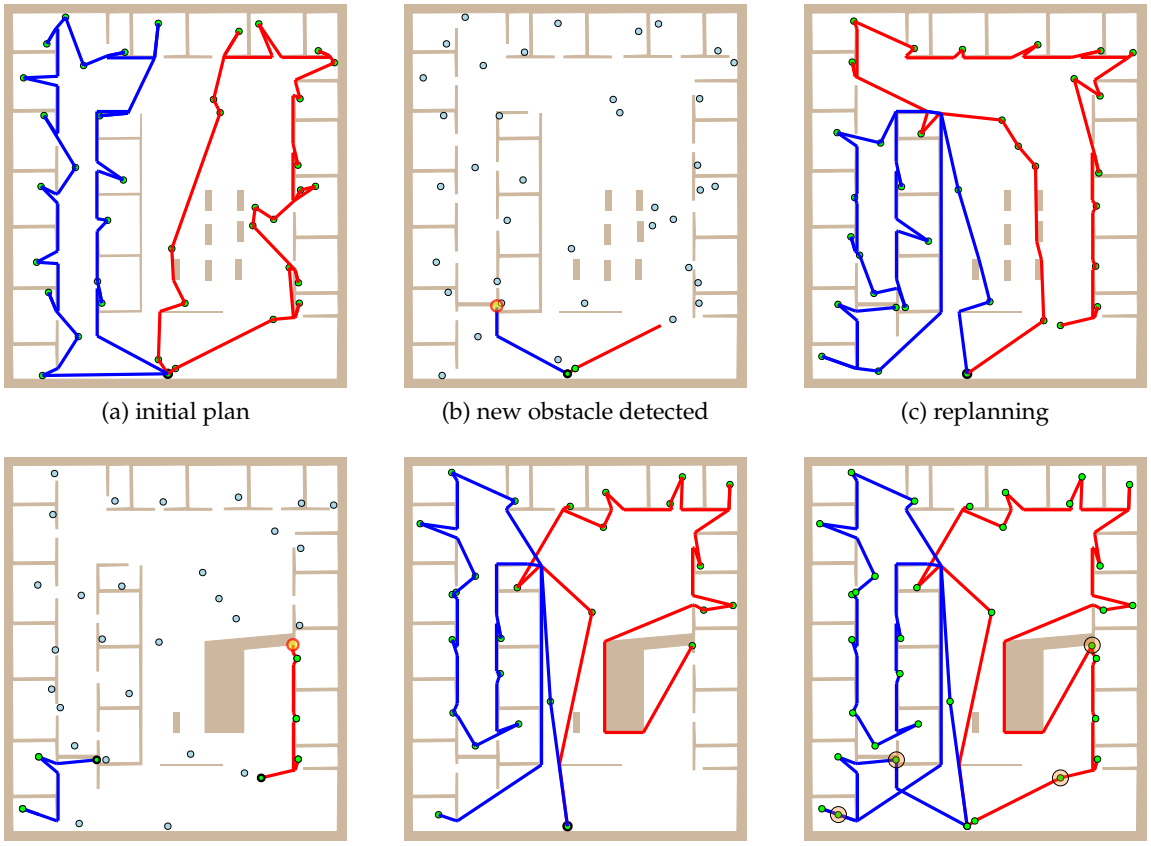
The MTSP formulation assumes a common depot. This condition is satisfied at the beginning of a search and rescue mission if all robots start from the common place. New obstacles can be detected during the plan execution, therefore the whole plan should be replanned. In such a situation, it is necessary to consider the multi-depot variant of the MTSP, because robots are spread around the environment.

The extension of Algorithm 5 is straightforward. The path (ring) has to start from the particular initial position of the robot and have to end at the common depot. The path is not closed, therefore the first node of the ring is adapted to the particular initial position, while the last node of the ring is adapted to the final common depot. Nodes represent linear string and neighbouring nodes at the ends of the string are considered only in particular direction. An example of the algorithm performance is shown in Figure 5.13.

5.5.2 Path Planning for Cooperating Heterogenous Entities

In a search and rescue mission with a team of heterogenous entities, some parts of the environment can be accessible only for a robot (entity) with special capability. In such case, it is necessary to consider different capabilities of particular robot and plan a path for the robot only inside the allowable regions. An example of such situation can be a team of cooperating robots and human rescuers. A path for the rescuer must be safe, hence it is necessary that the path avoids dangerous areas, where explosion can impend [161].

To satisfy these requirements a specific map, where restricted areas denote obstacles, is created for each type of the entity. During the adaptation phase winner nodes are selected according to a path in the particular map. Algorithm 5 needs to be extended to consider a specific map for each salesman. An example of solution is shown in Figure 5.14, the green path avoiding dangerous areas (shown in red) is for the human rescuer, while the blue path is for a robot, which can enter into dangerous parts of the environment.



(a) initial plan (b) new obstacle detected (c) replanning

(d) additional obstacle (e) next replanning (f) final travelled paths

Figure 5.13: An example of the path planning and replanning in the multi-depot MTSP variant.

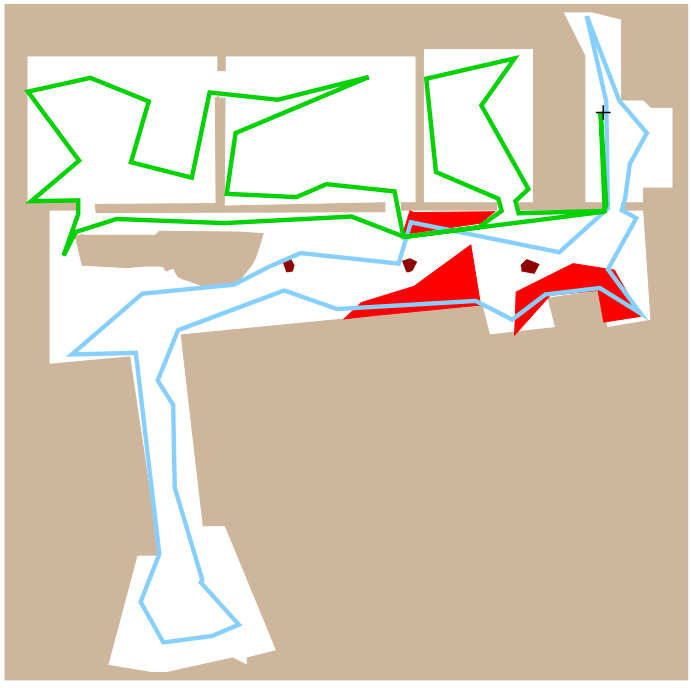


Figure 5.14: An example of MTSP solution for heterogenous entities.

5.6 Experimental Results

The proposed extensions of the SOM algorithm for the MTSP in a polygon domain have been experimentally verified in a set of problems. Due to lack of common problems for environments with obstacles a set of environments used in the motion planning problem has been utilized, parameters of the environments are shown in Table B.1.

The SPM approach does not provide sufficient robustness for the environments. Therefore the SOM algorithm has been evaluated in a set of TSPLIB problems [221] to estimate algorithm's performance prior evaluation of the shortest path approximation, the results are presented in Table B.10. The recommended parameters of the SOM procedure [243] have been used. A summary of the parameters is in Table 5.2. The adaptation process has been terminated if the $error$ is less than 0.001 or after 180 adaptation steps.

Found solutions are compared with exact solutions found by the Concorde solver [16] in the TSP case and with reference solution of the MTSP-MinMax found by GENIUS-*quality* as the best solution from 20 found solutions. The quality metrics follow notation described in Section 5.4, but ratios are used rather than absolute values. SOM and GENIUS are randomized algorithms, and therefore each problem has been solved 20 times by the particular algorithm variant and the average values have been computed. A summary of the notation is depicted in Table 5.3.

m	...	the number of salesmen.
$\#s$	...	the average number of adaptation steps.
LR	...	the average length ratio $\mathrm{L}/\mathrm{L}_{best}$, where L is the length of the longest tour and L_{best} is the length of the longest tour of the reference solution.
TR	...	the average time ratio, particular required computational time divided by the time of the reference solution.
CQ	...	the average value of the CQ (it is relative for the particular solution).
CER	...	the average *Collaborative Effort* ratio, it is computed as ratio of the particular CE and the value of CE for L_{best}.
s_x	...	the root of sample variance of the variable x.
MS	...	the percent of iterations, which are terminated by the maximal number of adaptation steps, of all iterations for the particular set of problems and the algorithm variant.
PDM	...	the percentage deviation to the reference (optimum) tour length of the mean solution value, $PDM = (\overline{L} - L_{ref})/L_{ref} \cdot 100\%$.
PDB	...	the percentage deviation to the reference of the best solution value.
No. Iter	...	the number of performed experiments.

Table 5.3: Used notation.

The examined problems are organized into three sets according to the number of cities: *small*, *middle* and *large*, see TableB.2. Particular quality of solution is examined over all solutions of the set. The cities are sensing locations found by the RDS and BP algorithms for the given visibility distance. Due to small number of cities in the *small* problem set only the *middle* and *large* sets are examined for the MTSP. The problems are created from the TSP variants by explicitly placed depot close to the center of the free space. In addition, for the *warehouse*, *jh*, and *h2* maps a depot is also placed near to the entrance, and therefore two problems for each environment are distinguished by subscripts A and B. Depots' positions (in map coordinates) are presented in Table B.3.

Three types of the SOM algorithm variants have been examined. The refinement of the shortest path approximation is studied in the TSP. Determination of the ring length is studied in the MTSP. In addition, for both problem variants the algorithm performance for alternative stop condition, Section 5.2.5, is compared with the original stop condition. The alternative stop condition is denoted as *unique* algorithm variant, and *error* represents the stop condition based on the minimal distance of the winner node to the city. In both variants, the maximal number of adaptation steps has been set to 180. The overall number of performed experiments is 3800 for the TSP and 5760 for the MTSP, where the number of salesmen m is from the set $\{2, 3, 4, 5\}$.

5.6.1 Refinement of the Shortest Path Approximation

The refinement of the shortest path approximation has been introduced in Section 5.2.3. The quality of path refinement has been studied for the node–city paths and evaluated in the three TSP problem sets. The rough path approximation over a cell vertex is denoted as *va-0*, and *va-k* is a refinement up to k vertices, the full path refinement is denoted as *pa*. Aggregated results for each problem set are presented in Table B.4, where selected time reference is for the *unique* variant with the *va-0* refinement, because it is expected to be the fastest one. The length ratio is determined from the exact solutions found by Concorde.

An error of the *va-0* variant caused termination of the adaptation after 180 steps in eight environments. The *va-0* refinement variant provides the worst overall the solution quality, but it is the fastest variant. The solution quality of the *va-1* variant is similar to the full path refinement and both variants are competitive with the Euclidean distance in the TSPLIB problems. Detail results are presented in Table B.5, where L_{opt} is the optimal solution found by the Concorde. Notice the computational time for the problems $h2_2$ and ta_1, the *h2* map contains more vertices, therefore it is more computationally intensive. Selected best found solutions for the *pa* variant are shown in Figure B.1.

The main results of the performed experiments is that the *va-1* approximation is sufficient and full path refinement does not provide significantly better solutions.

5.6.2 Determination of the Ring Length

The MTSP-MinMax SOM algorithm uses a length of the ring to prefer selection of nodes from shorter rings. Two variants of length of the ring, see Section 5.2.4, have been examined, the *nn* variant is based on the approximation of the shortest path between two nodes. The second variant, called *cc*, uses a length of the city tour represented by the ring. The full path refinement (*pa*) has been used for node–city paths in the performed experiments. Experimental results are presented in Tables B.6 and B.7, where the reference value of TR is the *nn* algorithm variant with the *error* stop condition. The reference solution of the MTSP is the best solution found by the GENIUS-*quality* algorithm from 20 solutions. Detail results for three salesmen are presented in Table B.8.

The proposed *cc* determination of the ring length outperforms the *nn* variant in the quality of solution and also in the required computational time. However the LR is lower about units of percents for *cc*, the CER is almost identical for both variants. Also notice CER is less than one. The CQ increases with the number of salesmen, which indicates higher differences in the individual lengths of tours for the particular solution. The proposed path approximation provides better results than the Euclidean distance for the selected TSPLIB problems, see Table B.10. It can be caused by obstacles, which constrain

the problem, therefore less combinations are "reasonable" and better solutions are found with higher probability.

Regarding the experimental results the proposed *cc* variant outperforms the original SOM algorithm.

5.6.3 Alternative Stop Condition

The alternative stop condition, the *unique* algorithm variant, has been experimentally evaluated with the TSP and MTSP experiments. In the case of the TSP, the solution quality as the average length is not affected, however the standard deviation is higher. The average number of required adaptation steps is lower than for the *error* variant as it has been expected. The inhibition mechanism is not used for the *unique* variant, thus if the algorithm does not converge in the defined number of steps, complete tour have to be determined.

For the *middle* problem set, the convergence issue is only minor and overall required computational time is about thirty percent lower than for the *error* variant. In the case of the *large* problem set, differences in the required computational time are not significant. Due to the convergency issue, higher number of adaptation steps is required and the adaptation is terminated after 180 steps, which leads to higher overall required computational time. The solution quality for the MTSP is similar for both variants *error* and *unique*, but for higher number of salesmen and problems from the *large* problem set the convergence fail in more than twenty percents.

The *unique* variant is not suitable for *small* problems, because found paths are about ten percent longer while the required computational time is about thirty percent lower that practically means tens of milliseconds. For the *middle* and *large* problem sets the quality of solution is not affected by the *unique* stop condition. The required computational time is also about thirty percents less for *middle* problem set whereas differences between *error* and *unique* stop conditions are negligible in the case of *large* set.

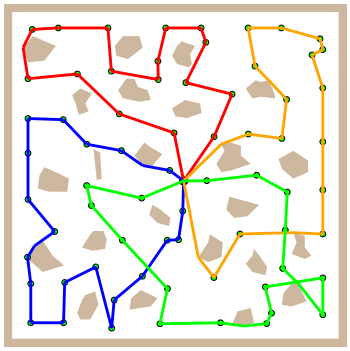
The alternative stop condition can be advantageous for the *middle* problems, a solution can be found in less computational time, while quality is not significantly worse (less than one percent). In all other cases, the regular stop condition (*error*) provides better performance.

5.6.4 Algorithm Comparison

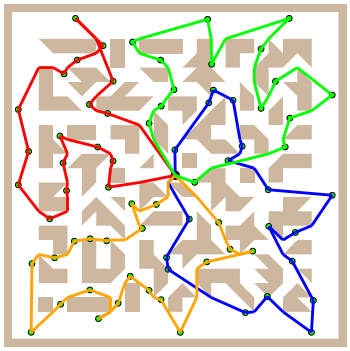
Results presented in Tables B.6,B.7 and B.8 provide overall comparison of algorithms. Regarding the value of LR the GENIUS algorithm provides overall better solutions with respect to the MinMax criterion. For the *large* set the SOM-*cc* variant provides similar LR like GENIUS-*quality* and significantly better results than GENIUS-*fast*. The GENIUS algorithm also provides better results from CQ point of view, which is for each problem less than one percent, that is significantly less than for SOM with several percents. It means that all tours are almost equally long in particular solution. The CER is lower for SOM, which is mainly due to fact that GENIUS improves only the longest tour.

For several problems the SOM algorithm provides shorter the longest tour than the best solution found by the GENIUS-*quality*, see Table B.8. The results also indicate that the SOM algorithm seems to be sensitive to the depot position, because quality of found solutions for depot A and B variants is significantly different, which can be related to

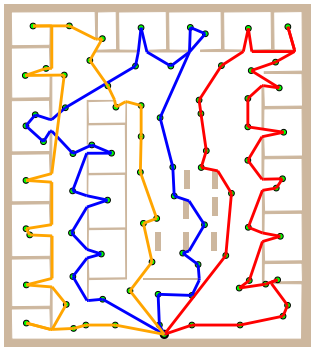
the important aspect of the SOM solutions. SOM tries to preserve topology of the input space, which leads to prefer solutions without mutually crossings tours. To illustrate this behaviour four best solutions found by both algorithms are presented in Figure 5.15.



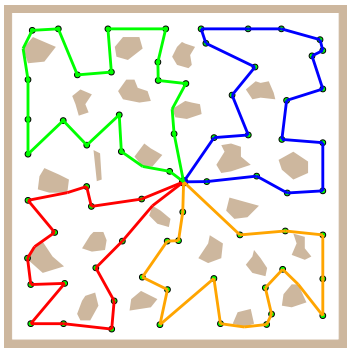
(a) GENIUS, L=46.3 m, CQ=1.5 %, CE=182.7 m



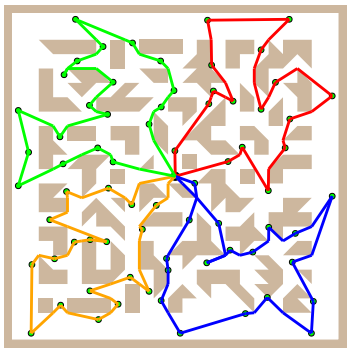
(b) GENIUS, L=11.7 m, CQ=1.6 %, CE=46.1 m



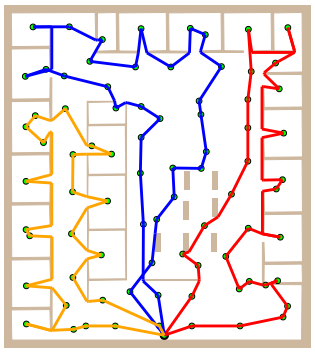
(c) GENIUS, L=80.5 m, CQ=0.4 %, CE=240.3 m



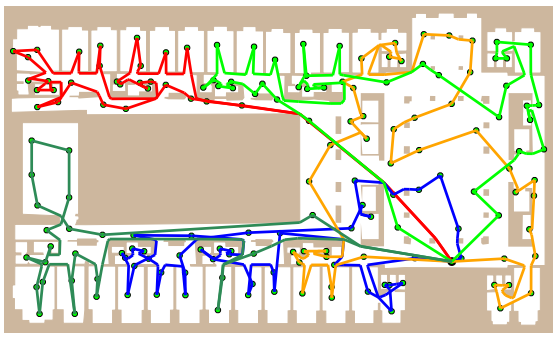
(d) SOM, L=46.1 m, CQ=6.0 %, CE=172.8 m



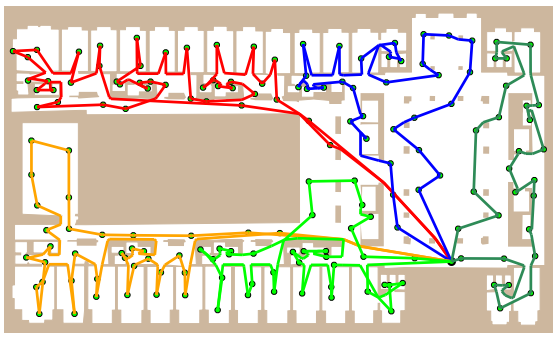
(e) SOM, L=11.6 m, CQ=3.4 %, CE=44.8 m



(f) SOM, L=81.7 m, CQ=5.1 %, CE=231.2 m



(g) GENIUS, L=253.06 m, CR=0.3 %, CE=1261.9 m



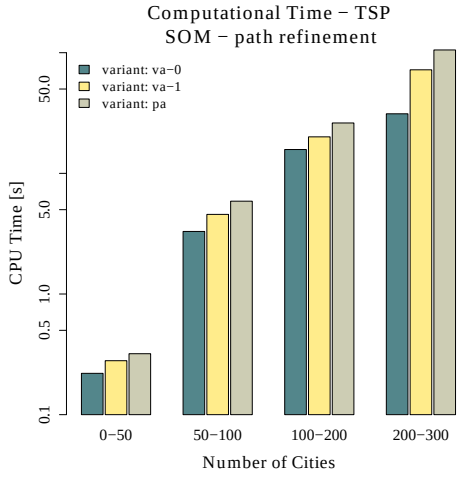
(h) SOM, L=275.4 m, CR=13.6 %, CE=1147.0 m

Figure 5.15: Examples of found solutions by the GENIUS-*quality* and SOM (*error*, *cc* variant) algorithm, L – length of the longest tour, CR – *Cooperative Ratio*, CE – *Collaborative Effort*: (**a,d**) map *potholes*, m=4;(**b,e**) map *m3*, m=4;(**c,f**) map *jh*, m=3;(**g,h**) map *h2*, m=5;

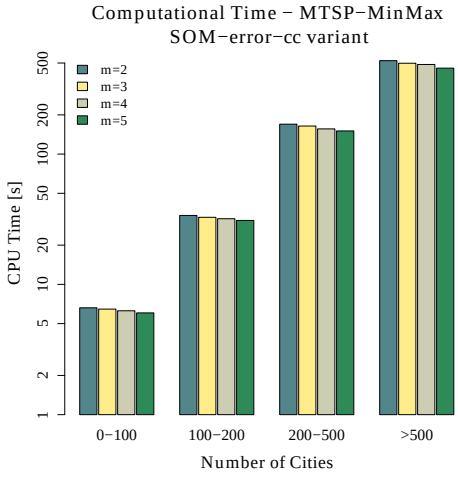
From the path planning point of view SOM provides an interesting feature, found tours tend to do not cross that means such solutions guarantee the coordination. To guarantee coordination for the GENIUS solutions it is necessary to consider velocity profiles for each robots and probably the robots will not collide, at least for the presented cases.

5.6.5 Required Computational Time

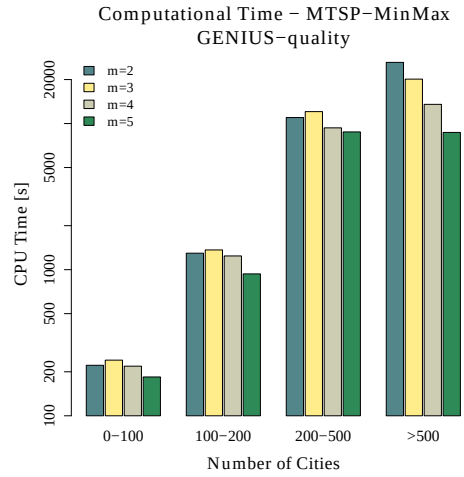
Two supporting structures have to be precomputed in order to use the proposed approximation of the shortest path. Required computational time to create a convex polygon partition is in units or tens of milliseconds and it is negligible in comparison to the required time of the adaptation procedure. Also a construction of the visibility graph is very fast, it is found in 41 milliseconds for the largest problem with 575 cities. The most time expensive part of the preparation phase is a computation of the shortest path between cities (and vertices), the time is included in the presented results. The shortest paths are also necessary for the GENIUS algorithm, and also the time is included in the presented results. The computational time depends on the number of cities and also on the particular environment, therefore the times are presented as histograms of average values in Figure 5.16. All algorithms have been implemented in C++ and compiled by the G++ 4.2 with -O2 optimization flag, all computations have been performed within the same computational environment using a single core of the Athlon X2 at 2 GHz CPU, 1 GB RAM, running FreeBSD 7.1.



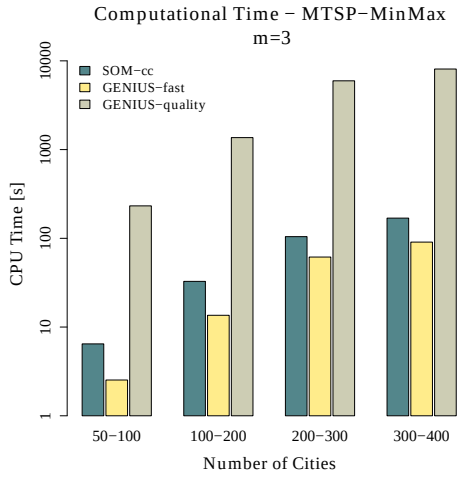
(a) TSP, SOM with *error* stop condition



(b) SOM-*cc* variant with *error* stop condition



(c) GENIUS-*quality*



(d) SOM-*cc* variant with *error* stop condition

Figure 5.16: Required computational time of the TSP/MTSP algorithms.

5.6.6 Adaptation on a Triangular Mesh

The adaptation procedure for a graph input has been evaluated for two problems with 36 and 356 cities and the *jh* map. A set of triangular meshes has been generated according to specified maximum triangle area and minimal triangle angle [237]. For each problem and particular mesh 20 solutions have been found and compared with optimal solution as percentage deviations PDM and PDB in Table B.9. In addition, solutions found by the proposed approximation of the shortest path are presented in the last three rows, the *error* stop condition is used. Examples of solutions are depicted in Figure B.2.

For the lower number of cities, the mesh is competitive with the convex polygon partition. In the examined problems, the *va-1* variant outperforms all other algorithm variants. The path refinement *va-0* provides solution with similar quality, but lower required computational time. The convergence trouble does not appear in the examined problems.

For the high number of cities the triangular mesh does not provide sufficient the solution quality, also it is more computationally intensive than the approximation of the shortest path. SOM on a triangular mesh is suitable for problems with lower number of cities, where lower number of triangles is sufficient.

5.7 Discussion

The proposed approximation of the shortest path allows application of the SOM adaptation procedure to the multi-goal path planning problems in a polygonal domain. A path based on the convex partition provides sufficient approximation, it is computationally feasible and it is not affected by geometrical issues like the SPM approach. Moreover, experimental results indicate that the full path refinement is not necessary.

The convex polygonal partition is one of the main advantage of the proposed approximation. It is a property of the environment, so it is independent on the set of cities (sensing locations). The proposed algorithm has been performed more than five thousands times without numerical issues, which indicates sufficient robustness of the algorithm.

Also experimental results show that the proposed SOM-*cc* variant provides better solution than the direct computation of the length of the ring. SOM-*cc* does not require approximation of the shortest path between two nodes, therefore only the node–city path queries are performed. The quality of found solutions is competitive with the general heuristic GENIUS, but solutions require less computational time than GENIUS-*quality*. The GENIUS-*quality* algorithm is very computational intensive, while GENIUS-*fast* is faster than the proposed SOM-*cc*, but SOM-*cc* provides better trade-off between the quality and required computational time.

Determination of the path among obstacles is more than hundred times computationally expensive than usage of the Euclidean distance. The initial version of the SOM approach based on computation of the visibility graph was very slow, even slower than the GENIUS algorithm [262]. The approximation of the shortest path significantly reduces the required computational time, while the quality of found solution is not decreased, according to the comparison of the path refinements and the SOM performance in the Euclidean TSP. From a real applicability in mobile robotics point of view the required computational time seems to be acceptable for problems with hundreds of cities, which is the typical number of cities in the inspection planning for maps of real buildings.

The used SOM schema [243] can be extended in various ways with respect to recent variants of SOM for the TSP, and it will be more likely outperformed by the memetic approach [64] or the Co-Adaptive Net [56]. The proposed modification of the adaptation procedure can also be applied in new SOM variants, because the main principle of SOM is the same. It should be remarked that the quality of the MTSP solutions is better for problems with obstacles than without them in comparison to solutions found by GENIUS. The quality can be related to the fact that the examined problems represent instances of the inspection task, where cities are placed in a planar map of an indoor environment. This observation is particularly important in the context of algorithm benchmarking and their real application, like in the presented multi-goal path planning in the inspection task.

The fastest proposed variant *va-0* of the path refinement does not provide sufficient quality of the path to find a stable solution of SOM in particular environments, but the adaptation can be terminated after a given number of adaptation steps and the best found tour can be used as the solution like in the Co-Adaptive Net algorithm.

A supporting graph instead of a convex partition can be used, especially for small number of cities. For higher number of cities (several thousands) the used triangular mesh provides worse performance than the proposed path approximation. The mesh contains many unnecessary vertices and edges, which are not used during the adaptation procedure. Possible future work on supporting structures can combine the triangular mesh and paths in the convex partition, because nodes are principally moved along the shortest path between cities (vertices). The important aspect is a discretization of the free space in order to provide sufficient domains for weights of neurons.

The SOM procedure also provides sufficient flexibility to solve particular problem variants that has been demonstrated in Section 5.5. Together with proposed extensions it represents a flexible tool to address multi-goal path planning problems for known or partially known environments, especially with concerns to following aspects:

- a single robot and optimization criterion (TSP formulation),
- a cooperating group of robots and optimization criterion (MTSP-MinMax formulation),
- a heterogenous team of robots,
- multi-depot variants.

The multi-goal path planning problem needs a determination of the path among obstacles, which can be very computational demanding. In [226], authors proposed reduction of the number of required paths, thus from this point of view, the SOM algorithm requires more than necessary number of paths and fast and efficient determination of the path is crucial for reasonable computational requirements. On the other hand, the SOM procedure provides interesting features, mainly the non-crossing paths that guarantee the coordination and simple competitive rule for the MinMax variant. To remember the related work, the coordination has been explicitly addressed for the two robotic arms by a modified competitive procedure in [215], which can be possibly applied in the MTSP. From this perspective the proposed SOM algorithm forms a solid framework for possible extensions to address the coordination issue or other robotics tasks, while it support relatively cheap solution of the cooperation according to the MinMax criterion.

The proposed approximation and experimental results show one possible way how the SOM procedure can be utilized in the multi-goal path planning problem and it can open possible future applications, e.g. for 3D environments, where approximate shortest paths are necessary.

Chapter 6

Generalized Multi-Goal Path Planning

The multi-goal path planning problem is formulated as the TSP in this thesis. If particular goals are partitioned into groups, the problem can be called *generalized multi-goal path planning problem* [226]. The problem is to find a shortest path to visit at least one goal from each group. In this chapter, the generalized multi-goal path planning problem is considered in the context of the inspection planning for a group of cooperating mobile robots. At first, an extension of the point sensing location to the segment is considered in the decoupled approach of the inspection task, thus the inspection task is accomplished if a robot performs measurement at least at one point of each segment, Section 6.1. After that, more general problem is examined, the goal is considered as an Area of Interest (AoI) that can be represented as a polygonal region in a polygonal domain, Section 6.2.

A solution of the proposed generalized problems is based on the SOM adaptation procedure described in the previous chapter. The main idea of the competition is same, but the adaptation rule is modified in order to adapt nodes towards the generalized goal. Due to lack of alternative approaches the proposed algorithms have been experimentally verified in selected problems, therefore instead of detail results (like in the previous chapters) only illustrative solutions are presented to demonstrate feasibility of proposed algorithms. The chapter is concluded with discussion of proposed extensions and further applications.

6.1 Inspection Planning for Segment Sensing Locations

A segment sensing location represents a set of possible locations where a measurement can be performed. The SOM procedure may adapt a node towards the segment goal, however it is necessary to find a path from a point to the segment. Determination of such a path can be supported by the SPM for the segment goal, which can be found by propagating a wavefront from the segment. The propagation starts with the segment wavefront parallel to the segment goal, and after the first event, the process is identical to the point goal. The path can also be found by the navigation function that provides a direction to the goal for a robot at any position inside the free space and the goal will be reached if the robot follows the direction [52]. The artificial potential field (APF) technique can be used to find suitable navigation functions, because it is able to generate an attractive potential

for an arbitrary shape, hence solution for the segment goal is found by the same method like for the point goal.

A set of segment sensing locations is found as a solution of the AGP with segment guards. The algorithm to find the set is described in Section 6.1.1 and the APF technique is briefly introduced in Section 6.1.2. An application of paths from the navigation functions in the MTSP is demonstrated in Section 6.1.3. Modifications of the SOM algorithm from the previous chapter for navigation functions and segment goals are described in Section 6.1.4.

6.1.1 Art Gallery Problem with Segment Guards

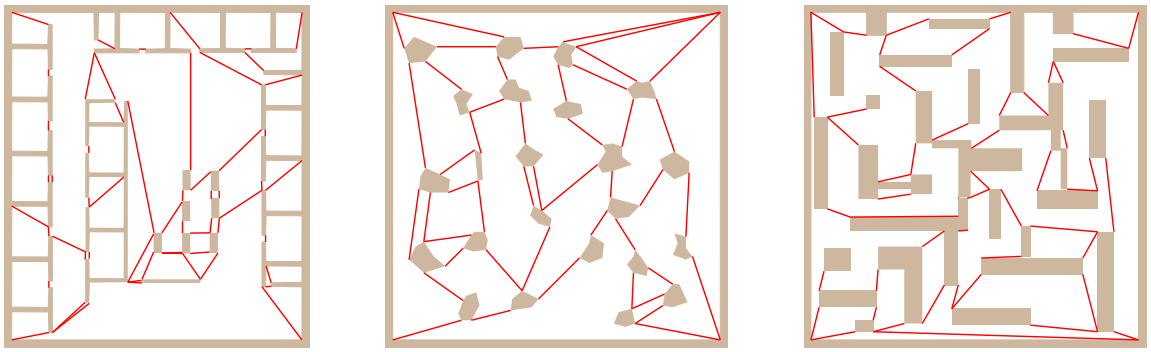
The segment guards represent sensing locations in the inspection task such that the whole environment is inspected if measurements are performed at each found location. The segment guard has been used by Toussaint as mobile guards and diagonal guards have been studied by Shermer [235]. To demonstrate the SOM approach for the segment goals, it is not necessary to have the minimal set of guards, thus a straightforward solution based on diagonals of polygonal representation of the robot workspace $\mathcal{W}$ is used. From the same reason only the case for unrestricted visibility range is considered.

Algorithm 7: AGP with segment guards

Input: $\mathcal{W}$ - workspace to be covered
Input: $\boldsymbol{C} = \{P_1, P_2, P_n\}$ - convex polygon partition of $\mathcal{W}$, P_i is a convex polygon
Result: $\boldsymbol{G} = \{d_1, d_2, \ldots, d_m\}$ - found set of segment guards

while $|\boldsymbol{C}| > 0$ **do**
  $d \leftarrow$ select random diagonal from $\boldsymbol{C}$ `// $P_i$ has at least one diagonal`
  $\boldsymbol{C} \leftarrow \boldsymbol{C} \setminus \{P_i | d \in P_i\}$ `// cover incident polygons with` d
  $\boldsymbol{G} \leftarrow \boldsymbol{G} \cup \{d\}$

The segment guards of the polygon $\mathcal{W}$ are found from the convex polygon partition [233] by Algorithm 7. The algorithm selects a random diagonal from the partition until the whole $\mathcal{W}$ is covered. The diagonal is incident with two convex polygons, thus two



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

Figure 6.1: Examples of segment guards.

polygons are covered from any point of the diagonal. A partition is found in $O(n \log n)$, where n is the number of $\mathcal{W}$ vertices. The number of diagonals is less than n, therefore

the total complexity of the algorithm is $O(n \log n)$. Examples of found segment guards are shown in Figure 6.1.

The presented Algorithm 7 can be easily extended for the restricted visibility range. Instead of simple convex polygon partition a convex cells used in the CPP, Section 4.3.1, can be utilized. The set of diagonals is replaced by the set of particular edges of the convex cells that are entirely inside $\mathcal{W}$.

6.1.2 Artificial Potential Field

The well known issue of the potential field methods are local extremes. A harmonic function has been used as the potential function [150] to avoid possible local extremes. A harmonic function g satisfies Laplace's equation $\nabla^2 g = 0$ on the domain Ω [57]. The boundary of Ω is denoted as $\delta\Omega$ and it represents the boundary of all obstacles in the configuration space $\mathcal{C}$. As consequence of the Gauss Integral Theorem the integral of g over the closed boundary is zero (6.1).

$$\int_{\delta\Omega} \frac{\partial g}{\partial \boldsymbol{n}} dS = 0 \tag{6.1}$$

The solution of Laplace's equation can be found analytically or numerically. The analytical solution can be found only for special cases and it is not affordable for general case, therefore numerical solutions are preferred. A solution can be found by the Finite Difference Method (FDM) [57], which uses a computation grid as an approximation of the partial derivatives by particular differences of neighbouring cells. The approach is suitable for the grid based maps, as information about surrounding environment is already represented in the grid form. The Finite Element Method (FEM), which is based on discretization of the solution domain, is suitable for the polygonal representation of environment. One of the common discretization is a triangular mesh, which can be created for constraints such as triangle angles or area.

A boundary condition may be specified to solve Laplace's equation. Two conditions can be considered: *Dirichlet* and *Neumann*. The Dirichlet condition specifies value of the function g at the boundary $\delta\Omega$

$$g = c(x, y), \tag{6.2}$$

for two dimensional configuration space, while Neumann condition specifies first derivatives

$$\nabla g = c(x, y). \tag{6.3}$$

Values of the boundary conditions $c(x, y)$ are usually constants and they are selected according to desired preference of clearance from the obstacles, the goal has the minimal value. The best practical approach is combination of Dirichlet and Neumann conditions, because if only Dirichlet condition is used for the boundary of obstacles, the following issue can occur. The value of the potential decrease from obstacles in direction to the goal. If a goal is too far from obstacles, a difference between two potential values of two neighbouring elements can be very small and usual numeric precision can be insufficient. Thus Dirichlet condition is applied to the goal and Neumann condition is applied for obstacles to increase value of potential around the obstacles.

For more details about FEM solutions see [103, 185] or applications in the trajectory control [89, 68] or path planning in the exploration task [78, 255] where authors discussed numerical methods to calculate potential fields based on specified boundary conditions.

(a) map *maze*, 15 808 elements, 29 000 triangles (b) map *var_density*, 19 243 elements, 35 942 triangles

Figure 6.2: Distance maps created from the navigation function for the point goal.

(a) map *maze*, 15 808 elements, 29 000 triangles (b) map *var_density*, 19 243 elements, 35 942 triangles

Figure 6.3: Examples of paths obtained by the navigation functions for the point goal.

The navigation function is represented by the value of potential of each element. These values can be used to construct the so-called distance map. The distance computed from the average potential of the triangle vertices can be associated to each triangle. Examples of the distance map are shown in Figure 6.2 where the dark blue color denotes the goal and the red color the farthest place from the goal. Such visualization provides overview of distances along the potential gradient to the goal.

A path from an arbitrary point to the goal can be found from the navigation function with desired smoothness. The path can be found as a sequence of points by the following procedure. Potentials of neighbouring vertices are used to determine the gradient from a

point inside some triangle, the gradient defines direction to the goal. Then, a new point is found from the found direction and given sampling distance. The sampling procedure is repeated for the new point until the goal is reached. Examples of paths from several points of obstacle boundaries are shown in Figure 6.3.

An advantage of the APF technique (particularly FEM) is ability to find a navigation function for a goal with an arbitrary shape, only boundary conditions need to be specified. An example of the distance map and several paths for a polygonal goal is shown in Figure 6.4. The environment is modified map *dense*, in which one of the obstacle is used as the goal. The solution is found by FEM for 20 143 elements and 37 098 triangles.

(a) distance map (b) paths to goal

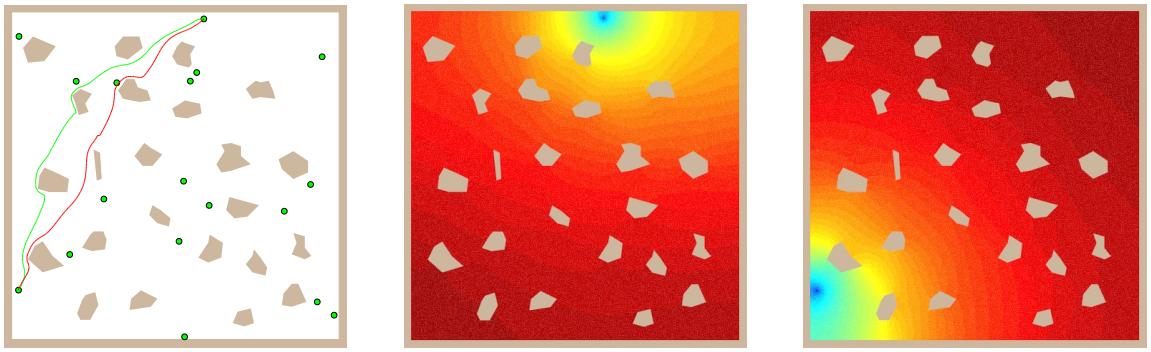
Figure 6.4: A navigation function for the polygonal goal, modified map *dense*.

All presented solutions of the APF in this chapter are found for the same settings of the boundary conditions. The value of Dirichlet condition (for the goal) has been set to -1 and the value of Neumann condition (for obstacles) has been set to 1. The solution of the APF is found as the solution of a system of linear equations. The required computational time to solve the system is in hundreds of milliseconds[1].

6.1.3 MTSP with APF Navigation Functions

The navigation function can be used to find paths between two cities in the MTSP formulated for a graph. Lengths of paths can define cost of edges. The navigation function is found individually for each city, therefore two paths can be used to connect two cities (two different APF problems are solved), see Figure 6.5. The green path corresponds to the navigation function for the upper city, while the red path is from the top city to the bottom city. Two possible paths leads to an oriented graph, or only the shortest path between two cities can be considered.

[1] The construction of the system of linear equations from elements of the triangular mesh takes several seconds using the Athlon X2 at2 GHz CPU, because of naïve implementation in Matlab 7.6. It is expected that a proper implementation in C will be much faster.



(a) two possible paths between the two point cities

(b) distance map for the green path

(c) distance map for the red path

Figure 6.5: An example of paths between two cities.

Once the graph is created, the multi-goal path planning problem with navigation functions can be solved by a TSP solver for a graph input, e.g. Concorde [16] or GENIUS described in Section 5.1.1. Examples of solutions of the TSP and MTSP-MinMax are shown in Figure 6.6. Cities are found as a guarding set and paths between cities are provided from the AFPs solved by FEM, therefore the found paths represent solution of the inspection task.

(a) TSP, d=*inf*

(b) MTSP-MinMax, d=2 m

Figure 6.6: Solutions of the inspection task with point sensing locations and navigation functions for the *potholes* map and the visibility range d.

6.1.4 SOM Adaptation Procedure for the Segment Goals

Paths for the segment goals can be found by the APF approach, but the graph based algorithms for the TSP cannot be directly used. The main issue is that a destination point at the segment depends on the position at the previous segment in the tour. If the position of a vertex (representing the segment) in the tour is changed, the path can be completely different, this is not an issue for point cities where changes are only local.

The issue is not the case of the SOM adaptation for the MTSP, because the tour is represented by the ring of nodes. A navigation function can be used similarly to the approximation of the shortest path described in Section 5.2.2. The winner node is selected according to its distance to the goal, which is determined from the path provided by a navigation

function. Such paths are also used to move nodes towards the goal. The adaptation procedure is terminated if each winner node is closer to the goal than the given distance δ.

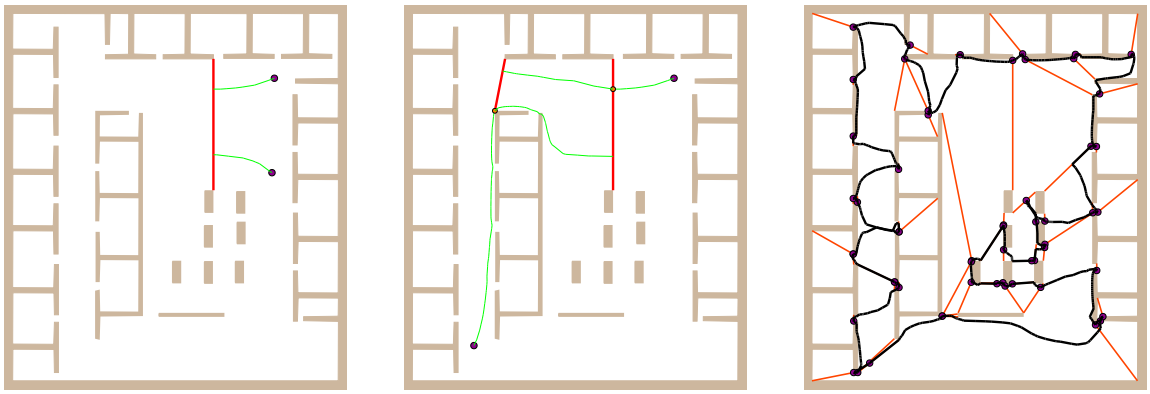
The final tour over goals (cities) can be easily constructed for the point goals, due to inhibition mechanism. In the case of the segment goals, an approximate tour can be constructed from the winner nodes, because they can be negligibly close to the segments. Alternatively a tour can be constructed by the following procedure, which can be also used for the MTSP-MinMax where nodes from shorter rings are preferred.

Assume two winner nodes ν_a and ν_b associated to the segments s_a and s_b, see Figure 6.7b for an example. Due to inhibition mechanism these two winners are distinct $\nu_a \neq \nu_b$. For each goal a navigation function is computed in advance. A path from the node ν to the segment goal s is determined by the navigation function $g_s(\nu)$ and the endpoint of the path at s can be denoted as $\mathrm{end}(g_s(\nu))$. Then, the goals s_a and s_b can be connected as follows.

1. Determine endpoints of paths from ν_a and ν_b to the segments s_a and s_b; $\vartheta_a = \mathrm{end}(g_{s_a}(\nu_a))$ and $\vartheta_b = \mathrm{end}(g_{s_b}(\nu_b))$.
2. Two paths connecting s_a and s_b can be constructed:
   (a) a path from ϑ_a to s_b defined by the $g_{s_b}(\vartheta_a)$ and a part of the segment s_b from the endpoints $\mathrm{end}(g_{s_b}(\vartheta_a)$ and ϑ_b,
   (b) a path from ϑ_b to s_a defined by the $g_{s_a}(\vartheta_b)$ and a part of the segment s_a from the endpoints $\mathrm{end}(g_{s_a}(\vartheta_b)$ and ϑ_a.
3. The shorter path from the two variants is selected and goals s_a and s_b are connected.

For a sequence of three goals s_a, s_b, s_c paths are firstly determined for each pair (s_a, s_b) and (s_b, s_c). After that, parts of the segments defined by the particular endpoints are added to the length of the path from s_a to s_c over s_b. The final tour over all goals is found by the same schema. This proposed procedure provides only approximate solution of the shortest path connecting given set of segment goals. The advantage of the SOM adaptation is that a winner node is very close to the end point at the associated segment.

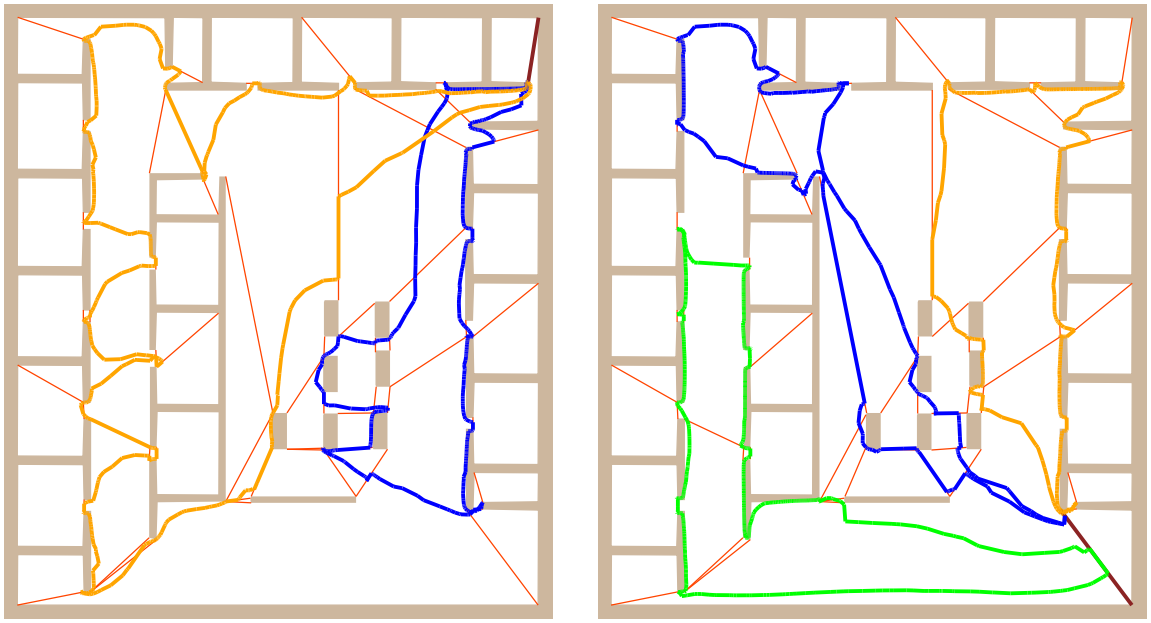
The SOM adaptation procedure with the navigation functions g_s is the same as for the Euclidean distance, the only difference is that instead of the Euclidean distance the navigation function is used to provide a path from a node to the city. A solution of the TSP is depicted in Figure 6.7c.



(a) paths to the segment goal (b) connections of segment goals (c) solution of the TSP

Figure 6.7: An example of found paths to segment cities.

The algorithm for the MTSP is similar to the TSP algorithm. A depot can be a point or a segment like other goals, the navigation function can be found for both variants. The length of the ring required in the competitive rule (5.2) is determined as the length of the tour represented by the ring. The procedure to find the final tour described above is used. Examples of MTSP-MinMax solutions are shown in Figure 6.8. The navigation functions are found by FEM with the average number of elements around three thousand and the average number of triangles around four thousands.



(a) two salesmen (b) three salesmen

Figure 6.8: Solutions of the MTSP-MinMax with 57 segment cities.

6.2 Inspection Planning for Areas of Interest

The next step of the goal generalization is consideration of polygonal region of the robot workspace $\mathcal{W}$. An example of such generalized problem can be the safari route problem introduced by Ntafos in 1992, which aims to find some shortest tour visiting given set of convex polygons attached to the inside boundary of a simple polygon [109]. The motivation for the safari route problem was the limited visibility range, because to see something interesting it is necessary to move closer to some region, but also it is not necessary to see the object of interest from a shorter distance than the visibility range d.

Therefore to see an object inside the convex polygon with size less than d it is sufficient to visit the border of the polygon. Such a polygon can be called *Area of Interest* (AoI) and its convexity guarantee that the polygon is covered (inspected) from any point of the polygon. The safari route problem can be too restrictive for a real application with mobile robots that is why the following extensions are considered in this section.

- At first, an AoI can be inside polygon with holes, which is more suitable model for an indoor environment than the simple polygon.
- An AoI can be entirely inside the free space, e.g. to inspect particular part of ceiling or floor, and not necessary attached to the boundary of the polygon.

- Particular AoIs can overlap each other.
- The cooperative behaviour of a group of mobile robots can be accomplished by the MinMax criterion, therefore similarly to the TSP also the safari route problem can be extended to find several tours with respect to the MinMax criterion.

According to the proposed extensions the studied problem can be formulated as follows.

> ***Path Planning Problem to Visit Areas of Interest*** - *Find m tours visiting given set of convex polygons (possibly overlapping each other) in a polygonal domain such that each polygon is visited from at least one tour, while the longest tour is minimized.*

The proposed problem is addressed by two approaches described in the next sections. In both approaches, the approximation of the shortest path, supporting triangular mesh or the APF can be used to determine a path from a node to the goal. The main ideas of the proposed approaches are independent on the path determination, therefore the approaches are presented for particular selected method. A discussion of the triangular mesh advantages are presented in Section 6.2.3.

6.2.1 Adaptation Procedure for the AoIs

The same SOM algorithm as for the segment guards can be used for the AoIs, because the suitable navigation function for the polygonal shape of the goal can be found by the same APF technique, see Figure 6.4 for an example. Also the final tour can be found by the same procedure like for the segment guards, due to convexity of the AoI. Another way (maybe more intuitive) is described in this section. It is based on the fact that an AoI can be represented by a set of sampled points. The problem can be then formulated as the MTSP and can be solved by the approach presented in Section 5.2. The main difference is in the selection of the winner node.

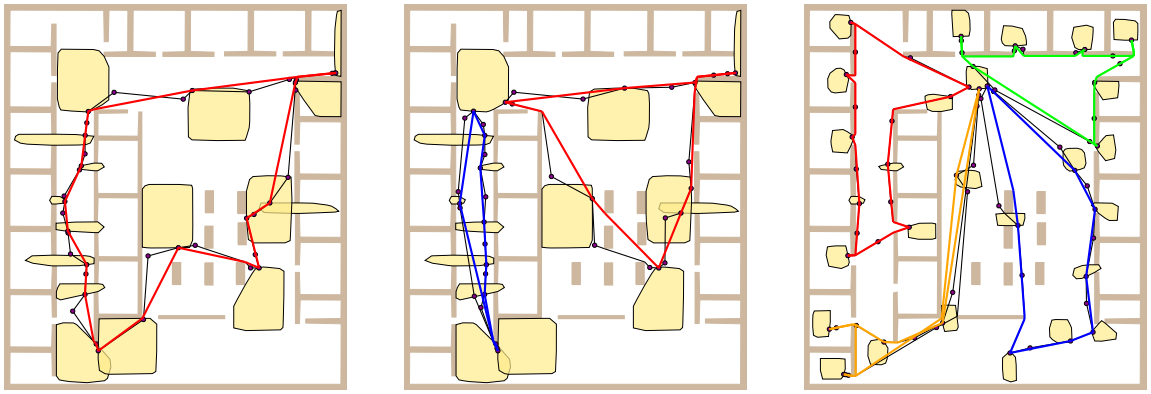
The closest node–point pair has to be found in the winner selection procedure of the SOM algorithm, because the goal is represented by a set of points. The competitive rule (5.2) is modified to consider selection of the best point from the set. Let the border of the polygonal goal G is sampled into finite number of points $G_{=}\{g_1, \ldots, g_k\}$. Then, the winner node is selected according to its minimal distance to G

$$\nu_{winner} = \mathrm{argmin}_{\nu \in \mathcal{N}, g \in G} |\nu, g| \cdot K, \tag{6.4}$$

where $\mathcal{N}$ is the set of nodes and K is the weight to prefer selection of winner from shorter ring introduced in (5.2). The length of the ring can be computed directly from an approximation of the shortest path between two points or as the length of the city tour represented by the ring.

The complexity of the presented approach depends on the number of goals and the number of sampled points. Thus, from this point of view it is more computationally intensive than the application of the navigation functions, which does not depend on the number of sampled points. The supporting structures for an approximation of the shortest path (node–city) must be prepared for all points representing the AoI, alternatively the node–node path approximation can be used.

Three solutions are shown in Figure 6.9 where the black line segments represent rings of nodes. Particular winners, the closest nodes to the AoIs, are connected by the approximation of the shortest path to form the final tours, which are represented by the bold colored line segments. The solutions have been found in units of seconds using the Athlon X2 at 2 GHz CPU, FreeBSD 7.1 and C++ implementation compiled by the G++ 4.2 with -O2 optimization flag.



(a) m=1 (b) m=2 (c) m=4

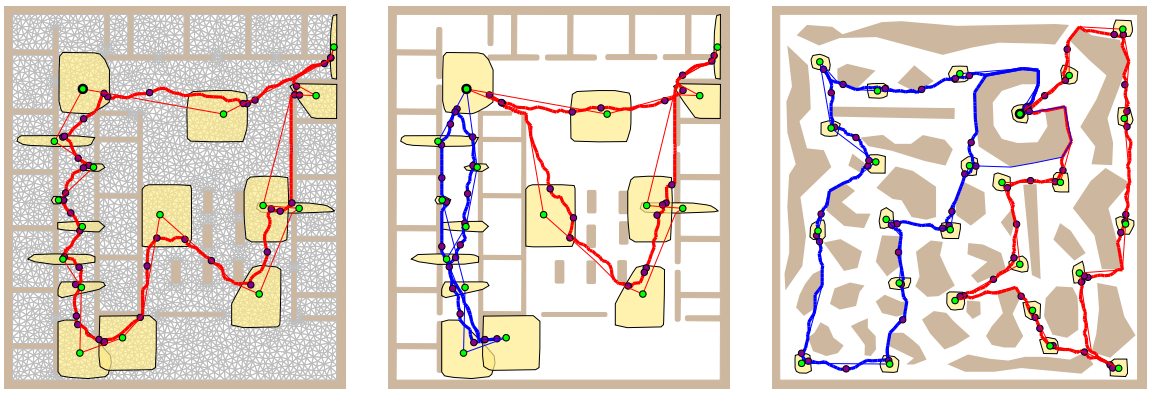
Figure 6.9: Inspection planning for AoIs, m salesmen and MinMax criterion.

6.2.2 Representative of the AoI

The main drawback of the above presented approach is sampling of the AoI shape by several points, which increases computational burden. An alternative approach can be based on the geometrical interpretation of the ring evolution and shape of the AoI. The idea is as follows: use one representative point of the AoI as an attractor for a node movement and if the node is inside the AoI stop its movement. The AoI is convex, thus it can be inspected from any point inside the area. A suitable representative point of the AoI can be the centroid of the convex polygon. The adaptation procedure for the MTSP, Algorithm 5, is modified to adapt nodes towards the centroid in the following way.

1. A winner node is selected according to its distance to the centroid of the presenting AoI to the network,
2. The winner node (and its neighbouring nodes) are moved towards the centroid only if they are not already inside the AoI.

Determination if a node (point) is inside the AoI with n vertices can be done in $O(n)$ by the winding number or in $O(\log n)$ [206], because of AoI convexity.



(a) map *jh* (b) map *jh* (c) map *dense*

Figure 6.10: Inspection paths for AoIs with representative of the AoI.

Three solutions found by the proposed modified adaptation procedure with representative of AoIs (and supporting triangular mesh) are shown in Figure 6.10. The thin colored

line segments represent the shortest paths between centroids of the AoIs and can be used for the final tour. The solutions have been found in hundreds of milliseconds in the same computational environment used for the experiments in the previous section.

6.2.3 Adaptation with Supporting Triangular Mesh

The triangular mesh approach can be computationally intensive for many triangles. On the other hand, it can be very efficient for lower number of triangles, and if particular paths to goals are precomputed and associated to each vertex of the mesh.

For the approach based on the AoI sampling, the most suitable point of the AoI border is selected in linear time. The selection process can be speeded up using pre-computed the best sampled point for each vertex of the mesh, thus the complexity will be the same as for the selection procedure of the TSP algorithm for the point cities, Section 5.3.

The triangular mesh can also be advantageous for the representative of the AoIs. Each vertex and edge of the mesh can have associated information about incident AoIs that can be used to speed up the process of the determination if a node is inside the AoI.

6.3 Discussion

The generalized multi-goal path planning problem has been addressed in this chapter by the SOM adaptation procedure. Two particular instances of the generalized problem have been introduced: cooperative inspection of segment goals and cooperative inspection of convex polygonal AoIs. Three approaches have been proposed to find a solution of these problems:

- adaptation of nodes using navigation function,
- selection of a point goal from the set of points representing the generalized goal, and
- adaptation to the representative of the AoI.

A solution of the problem with segment goals have been demonstrated in the inspection planning for segment sensing locations where point guards have been replaced by the segment guards. The key component of the proposed solution is application of a navigation function, which provides a path to more general goal than just a point. Navigation functions have been found as solutions of the harmonic APF by FEM. In general, any navigation function can be used, the only requirements of the SOM adaptation procedure are:

- it has to provide a distance from an arbitrary point in the free space to the goal, and
- it has to provide a path, along which a node is moved towards the goal, e.g. a sequence of points in a polygonal domain.

The main limitation of the navigation functions relates to the computational requirements. The functions can be precomputed, but for example in the case of the used FEM for the harmonic potential with several thousands of finite elements, the memory requirements increases quickly with every new goal. One of the possible future work can be based on utilization of new computational capabilities of the GPGPU [1] to find a solution of the APF in real-time. The used FEM method is common technique, therefore such general algorithms can be expected in near future, as the capabilities of widely used GPGPUs are going to be more and more powerful.

An important feature of the used APF should be remarked. The found paths are smooth, therefore a velocity profile for a path can be found, e.g. by the procedure [86]. Such a profile can be found with consideration of the maximal allowable velocities and accelerations, hence such planned paths can satisfy additional kinodynamic constraints. Moreover found velocity profile can be used to determinate cost of the path as the required time to travel. On the other hand, found paths (from certain points) are unnecessarily long, e.g. in Figure 6.3a. This issue is discussed in [185], where a solution based on a division of the free space is proposed.

In relation to the smoothness of paths, it should also be noted that connected paths (as a solution of the TSP) are not smooth, see Figure 6.6. This is not an issue for the discrete sensing in the related inspection task. A robot with the differential nonholonomic drive is assumed and the robot stops at the sensing locations to perform measurement and it is able to turn in the required direction and follows the planned path.

The selection of a point goal from the set of sampled points represents straightforward extensions of the SOM algorithms for the MTSP. The advantage of the method is that it does not require computation of the navigation functions, which can be more computationally intensive than the selection process for small problems.

The adaptation to the representative of AoI replaces the sampling of the polygonal goal by a simple test if a point is inside the convex polygon. The proposed adaptation rule shows advantage of the evolution of ring of nodes in a polygonal domain. In a certain sense, the rule demonstrates combination of visibility with metric properties in the so-called hybrid-visibility problems introduced by Shermer. This concept is extended in the next chapter where an adaptation procedure for the watchman route problem is proposed.

A path between node and presented goal to the network can be determined by the approximation of the shortest path, navigation function, or adaptation can be performed on a graph (triangular mesh). In general, any motion planning algorithm can be used, however for practical application it has to provide a path sufficiently fast to get reasonable computational time of the SOM algorithm.

The SOM adaptation procedure allows straightforward solution of the problem for several robots, the MinMax criterion is considered by preference of nodes from shorter rings, where a length of ring can be found as a length of the tour represented by the ring. The procedure allows to specify the common depot (starting point), or individual depots for a particular robot. These features together with adaption in a polygonal domain make the SOM adaptation procedure very flexible. The introduced path planning problem to visit AoIs is an extension of the safari route problem. Also the zoo-keeper route problem [50] can be solved by the proposed algorithms using the modification described in Section 5.5.2.

To conclude this chapter, the presented solutions of the proposed problems demonstrate flexibility of the SOM adaptation procedure to address various routing problems, which are related to the geometrical properties. The proposed modifications or more precisely supporting structures for the SOM can be eventually applied for another hard problems studied in the computational geometry, e.g. a solution of the problem with segment goals can be inspiring for the touring polygons problem.

Chapter 7

Inspection Planning with Cooperative Continuous Sensing

This chapter is dedicated to the problem of inspection planning with continuous sensing that can be formulated as the Watchman Route Problem (WRP) and as the Multiple Watchmen Route Problem (MWRP) with the MinMax criterion in the case of a group of cooperating mobile robots Similarly to the sensor placement problem the limitation of the sensing range is more realistic in the context of search and rescue mission, therefore the problem can be constrained to the d-visibility and a suitable variant of the WRP formulation becomes the *d-sweeper route problem*. Moreover size of mobile robots should also considered. Similarly to the previous chapters a point robot (with differential nonholonomic drive) can be assumed in the shrunk polygonal representation $\mathcal{W}$ of the robot workspace.

Regarding the additional constraints the studied problem of inspection planning with continuous sensing for a group of cooperating robots can be formulated as follows:*For a given polygon $\mathcal{W}$ (possibly with holes) find m paths in $\mathcal{W}$ such that all points of $\mathcal{W}$ are d-visible from at least one point of a path from the set of found paths, and the length of the longest path is minimized*. For short, the problem is referred as the MWRP-MinMax and as the WRP for the single robot in the rest of this chapter.

The formulated problem is addressed by a new proposed algorithm that is based on the representative of AoI introduced in the previous chapter, Section 6.2.2. The idea of the proposed adaptation procedure can be summarized in following observations.

- The SOM adaptation procedure evolves a ring of nodes in a geometrical space, and the ring "explores" topology of the free space during the learning phase.
- The ring can form a watchman route and a coverage of the route can be computed during the adaptation.
- Uncovered parts of the workspace $\mathcal{W}$ can be determined from the ring coverage.
- Representative points of uncovered parts of $\mathcal{W}$ can be used to attract nodes towards them, just like cities are presented to the network in the TSP.
- It is sufficient if the route just enters into a part of $\mathcal{W}$ for covering the part, thus representative points of uncovered parts are more like attraction points.

It is obvious that some kind of visibility have to be utilized during the adaptation to deal with the continuous sensing along the ring. Similarly to the problem with obstacles, the naïve approach is too computationally intensive, therefore it is necessary to use a supporting structure to decrease computational burden.

The realization of the proposed adaptation procedure can be decomposed into three sub-problems:

1. determination of the current coverage along the ring and uncovered parts of $\mathcal{W}$,
2. determination of attraction points of the uncovered parts,
3. adaptation of nodes to attraction points.

The first two sub-problems are addressed by supporting structure which is discussed in the next section together with an algorithm to compute a coverage of the ring. The proposed adaptation procedure and algorithms to solve the WRP and variants of the MWRP-MinMax are presented in Section 7.2. Experimental results are presented in Section 7.4. The last section of this chapter is dedicated to discussion and future work.

7.1 Supporting Structures and Algorithms

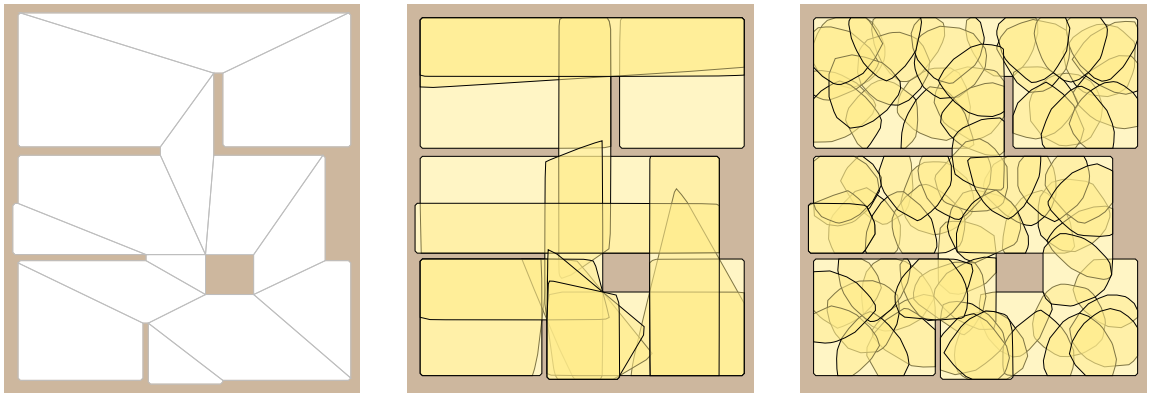
Convex polygons are advantageous in visibility problems. Just to remind one of the earliest work, Fisk's proof of the Art Gallery Theorem is based on supporting triangulation, which is a convex partition. A convex partition of a polygon P is a collection of convex sub-polygons within pairwise disjoint interiors whose union is exactly the polygon P. Another commonly used collection of convex polygons is a convex cover, in which sub-polygons can overlap. More formally a convex cover of polygon P is a collection of convex polygons $P_1, \ldots, P_k$ such that $P_1 \cup \ldots \cup P_k = P$.

Let's think about watchman walk in a convex partition and cover. If a watchman have to see whole free space of the polygon $\mathcal{W}$, it have to explicitly visit all polygons of the partition. It means while one polygon is visited the watchman has to move to another polygon and has to leave the current polygon. In a convex cover, a watchman can cover several convex polygons from a single point, because polygons can overlap. Hence to cover additional part of the environment smaller movement can be sufficient. From this point of view a convex cover seems be more suitable for the WRP algorithm, however it can contain more polygons than partition. Examples of convex partition and cover are shown in Figure 7.1.

The problem of finding the minimal convex cover is know to be NP-hard even for a simple polygon without holes [65], which is not helpful to decrease computational requirements. The intention of the convex cover is to supported determination of covered parts of $\mathcal{W}$ from a watchman route (ring of nodes) in such a way that incident convex polygons with the ring are the covered parts of $\mathcal{W}$. From this perspective any convex cover can be possibly used, the important aspect of the supporting structure is efficient determination of ring coverage and also determination of attraction points.

Regarding the requirements of the supporting structure a triangular mesh is used to find a convex cover, because it can be used to support determination of the current coverage of the ring. Moreover the centroid of each triangle can be used as an attraction point. The idea of the coverage computation is based on determination of incident triangles with the ring by the straight walk algorithm and association of convex polygon from the cover to each triangle. The current coverage of the ring can be computed by the union of all incident polygons, which can be efficiently implemented due to discretization of the free space into a finite set of triangles.

A triangular mesh can be found by the quality mesh generator [237] that provides a tri-



(a) convex partition (b) convex cover, d=inf (c) convex cover, d=1 m

Figure 7.1: Example of convex partition and convex cover of map *jari*, polygons are restricted to the size d.

angular mesh with specified maximum triangle area and minimal triangle angle[1]. Smaller triangles lead to higher number of triangles and to find several overlapping convex polygons, but with more triangles the computation becomes more intensive. Another aspect of the mesh quality is related to limited numeric precision and the walking procedure in the mesh. Triangles should be equilateral (close to be equilateral) or the Delaunay property must be satisfied, otherwise issues with degenerative cases can be expected [72].

More formally the triangular mesh $\mathcal{T}$ is a triplet $\mathcal{T} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{T})$, where $\boldsymbol{V}$ is the set of vertices, $\boldsymbol{E}$ is the set of edges $e \in \boldsymbol{E}$, $e = (v_i, v_j), v_i, v_j \in \boldsymbol{V}, v_i \neq v_j$, $\boldsymbol{T}$ is the set of triangles $T \in \boldsymbol{T}$, $T = (\{v_i, v_j, v_k\}, \{e_i, e_j, e_k\})$ where $v_i, v_j, v_k \in \boldsymbol{V}$, $e_i, e_j, e_k \in \boldsymbol{E}$, $e_i = (v_k, v_i)$, $e_j = (v_i, v_j)$, $e_k = (v_j, v_k)$.

7.1.1 Finding a Convex Cover

A convex polygon P of convex cover is found as a convex hull of mesh triangles, i.e. mesh of triangles' vertices. Each polygon P is formed from a sequence of vertices $\boldsymbol{V}_P$, $\boldsymbol{V}_P \subseteq \boldsymbol{V}$ and has associated set of mesh triangles $\boldsymbol{T}_P$, $\boldsymbol{T}_P \subseteq \boldsymbol{T}$ that are entirely inside the polygon $P, T \in \boldsymbol{T}_P, T \subseteq P$. The procedure to find a convex polygon P is depicted in Algorithm 8.

A construction of a convex polygon is started from a (possibly random) triangle T_r, which forms an initial convex hull. The hull is eventually extended by vertices that are opposite to outer edges of associated triangles to the polygon, set $\boldsymbol{E}_{open}$. Such vertex eventually extends the hull by one triangle. During the hull extension (convex_hull) the size of the hull is considered and vertex being added is eventually discarded. The procedure is repeated until list of candidate edges $\boldsymbol{E}_{open}$ is empty. An example of the algorithm performance is shown in Figure 7.2.

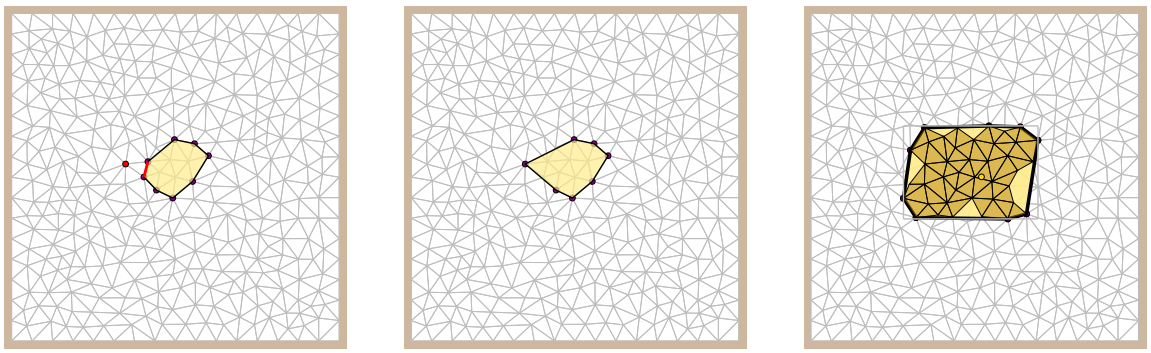
The algorithm to find a convex polygon is used to find a convex cover. The set of convex polygons is found by the randomized incremental procedure that selects a random uncovered triangle that is extended to a convex polygon. The procedure is depicted in Algorithm 9 and examples of found convex covers are shown in Figure C.1.

[1]The termination of the algorithm is guaranteed only for the angle 28.6° or smaller, however in practice higher values can be used.

Algorithm 8: Find Convex Polygon

Input: $\mathcal{T} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{T})$ - a triangular mesh of $\mathcal{W}$
Input: d - visibility range
Input: T_r - initial triangle, $T_r \in \boldsymbol{T}$
Output: $P(\boldsymbol{V}_P, \boldsymbol{T}_P)$ - convex polygon

$\boldsymbol{V}_P \leftarrow T_r(\boldsymbol{V}), \boldsymbol{T}_P \leftarrow \{T_r\}$ `// initial convex hull from` T_r
$\boldsymbol{E}_{open} \leftarrow \{e | e \in T_r\}$ `// add all edges of` T_r `to the open list`
$\boldsymbol{E}_{close} \leftarrow \emptyset$ `// initialization of the close list`
while $|\boldsymbol{E}_{open}| > 0$ **do**
  $e^\star \leftarrow \text{random}(\boldsymbol{E}_{open})$ `// select random edge from the open list`
  $T^\star \leftarrow T$ incident with $e^\star \wedge e^\star \in T(\boldsymbol{E}) \wedge T \notin \boldsymbol{T}_P$
  $v^\star \leftarrow v$ such that $v \notin \boldsymbol{V}_P \wedge v \in T^\star$ `// select possible candidate vertex`
  $C \leftarrow \text{convex\_hull}(\boldsymbol{V}_P, v^\star, d)$ `// try to extent the convex hull`
  if $T^\star$ *is entirely inside* C **then**
    $\boldsymbol{V}_P \leftarrow C, \boldsymbol{T}_P \leftarrow \boldsymbol{T}_P \cup \{T^\star\}$
    $\boldsymbol{E}_{open} \leftarrow \boldsymbol{E}_{open} \cup \{e | e \in T^\star(\boldsymbol{E}) \wedge e \notin \boldsymbol{E}_{close}\}$ `// add edges to` $\boldsymbol{E}_{open}$
  else
    $\boldsymbol{E}_{close} \leftarrow \boldsymbol{E}_{close} \cup \{e | e \in T^\star(\boldsymbol{E})\}$
  $\boldsymbol{E}_{open} \leftarrow \boldsymbol{E}_{open} \setminus \{e^\star\}$ `// remove edge from the open list`
  $\boldsymbol{E}_{close} \leftarrow \boldsymbol{E}_{close} \cup \{e^\star\}$ `// add edge to the close list`



(a) a vertex being added to the hull
(b) an extended polygon
(c) final convex polygon and associated triangles

Figure 7.2: An example of convex polygon determination.

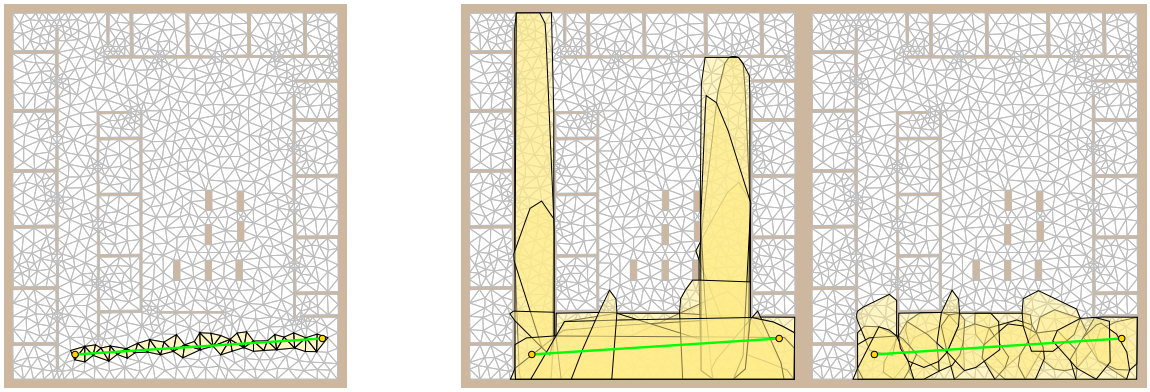
Algorithm 9: Find Convex Cover

Input: $\mathcal{T} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{T})$ - a triangular mesh of $\mathcal{W}$
Input: d - visibility range
Output: $\boldsymbol{P} = \{P_1, \ldots, P_n\}$ - set of convex polygons (convex cover of $\mathcal{W}$)

$\boldsymbol{U} \leftarrow \boldsymbol{T}$ `// uncovered triangles`
while $|\boldsymbol{U}| > 0$ **do**
  $T \leftarrow$ *select random triangle from* $\boldsymbol{U}$
  $P(\boldsymbol{V}_P, \boldsymbol{T}_P) \leftarrow \text{find\_convex\_polygon}(\mathcal{T}, d, T)$
  $\boldsymbol{P} \leftarrow \boldsymbol{P} \cup P(\boldsymbol{V}_P, \boldsymbol{T}_P)$
  $\boldsymbol{U} \leftarrow \boldsymbol{U} \setminus \boldsymbol{T}_P$

7.1.2 Determination of Coverage of the Ring

A convex cover of the workspace $\mathcal{W}$ together with a triangular mesh of $\mathcal{W}$ are utilized in finding a coverage of the ring. Each triangle lies at least in one convex polygon, therefore all such convex polygons are associated to each triangle. Nodes of the ring are inside $\mathcal{W}$, hence each node ν is at least inside one triangle. An approximation of the continuous sensing along the ring is based on the computation of coverage along a straight line segment s of two directly visible points. Incident convex polygons with a segment s are determined from the incident triangles, which are found by the visibility walk in a triangular mesh. An example of segment coverage is shown in Figure 7.3.



(a) incident triangles

(b) incident convex polygons, d=50 m

(c) incident convex polygons, d=4 m

Figure 7.3: An approximation of the continuous sensing along the line segment for the visibility range d, the triangular mesh contains 2266 triangles, map *jh*.

The coverage of a ring is determined in four steps.

1. A sequence of points $(p_1, p_2, \ldots, p_n, p_{n+1})$, where $p_1 = p_{n+1}$ for the closed ring, representing the ring is found by the approximation of the shortest path between each neighbouring nodes of the ring.
2. For each segment of neighbouring points $s_i = (p_i, p_{i+1})$ incident triangles are determined by the visibility walk in a triangular mesh. A set $\boldsymbol{T_r}$ is the union of all such incident triangles.
3. For each p_i all incident triangles are found[2] and added into $\boldsymbol{T_r}$.
4. The current coverage of the ring is a set of covered triangles $\boldsymbol{T_c}$ that is determined from the convex polygons. For each $T_i \in \boldsymbol{T_r}$ all associated convex polygons $\boldsymbol{P_i} = \{P_{i,1}, \ldots P_{i,n}\}$ are used to find the set $\boldsymbol{T_c}$,

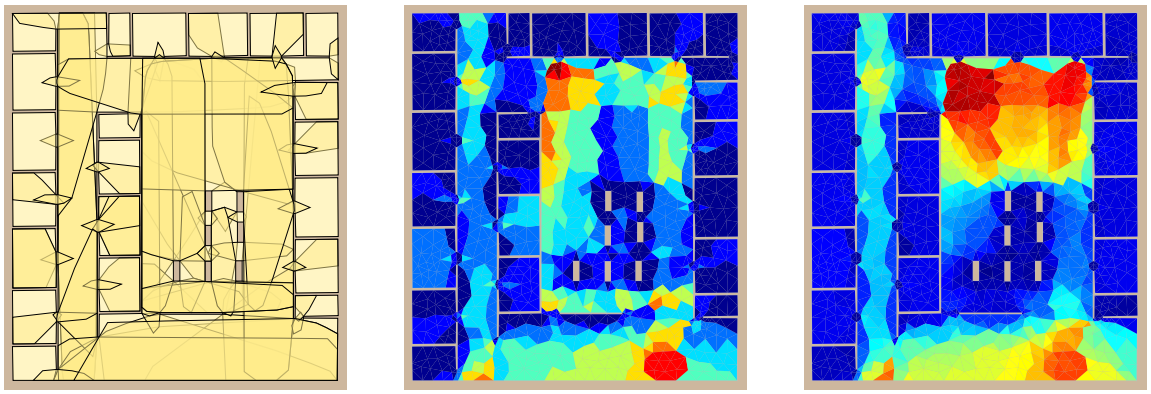
$$\boldsymbol{T_c} = \bigcup_{T_i \in \boldsymbol{T_r}} \bigcup_{P_{i,j} \in \boldsymbol{P_i}} \{T | T \in P_{i,j}\}. \tag{7.1}$$

Remark about Coverage and Continuous Sensing

A visualization of triangle coverage is shown in Figure 7.4, the color of a triangle indicates area of the covered part (area of union of all associated convex polygons). The highest cov-

[2] Necessity of this step depends on implementation of the walking procedure, because it can pass only triangles in the direction from p_i to p_{i+1}. In fact, all incident triangles can be found during determination of the first passed triangle in the direction.

erage is in red, while the smallest is in blue. However it seems naturally to place guards inside parts with the largest coverage (e.g. to find an approximate solution of the AGP), to cover the whole free space of $\mathcal{W}$ from the watchman route it can be better to prefer parts with smaller coverage. For the WRP with continuous sensing the colors should be interpreted with a different intention. The red parts will likely be covered from other parts, while the blue parts (rooms) have to be covered explicitly. In the context of the watchman route, red parts will be mostly covered from the path to visit the blue parts (rooms).



(a) (b) (c)

Figure 7.4: Visualization of triangles according to area of associated convex polygons, triangular mesh with 2266 triangles, unrestricted visibility range: (**a**) convex cover; (**b**) triangle coverage of a convex cover; (**c**) triangle coverage for the convex cover created from all triangles.

7.2 SOM Adaptation Procedure for the WRP

The proposed adaptation procedure for the WRP is based on a triangular mesh of $\mathcal{W}$ and a convex cover of $\mathcal{W}$, in which convex polygons have associated triangles of the mesh. The centroids of the triangles are used as attraction points similarly to cities in the TSP algorithm. The problem is to find a route to "see" all triangles, therefore it is not necessary to visit all triangles. The adaptation rule is modified to do not place a node unnecessary close to an attraction point p_a. An alternate point is determined and used instead of p_a if the node would be closer to p_a than the visibility distance d after the adaptation. A schema of the adaptation procedure is depicted in Algorithm 10.

The algorithm is similar to the adaptation procedure for the TSP, the main difference is in consideration of the coverage, all SOM properties (G, f, μ, α, k) are same. The current coverage is represented by a set of triangles $\boldsymbol{T}_c$. At the beginning of each adaptation step, a coverage of the current ring is determined. After that, triangles are presented to the network in a random order and nodes are adapted only for uncovered triangles. A winner node is selected according to its distance to the centroid p_a of the presented triangle T. The distance is found as a length of the approximate shortest path from the node to p_a. If a winner node is selected then triangles of all associated convex polygons $\boldsymbol{P}_c$ to the triangle T are added to $\boldsymbol{T}_c$, which means the ring is not adapted to these triangles in the current adaptation step. Also the adaptation is performed only if the winner node is not

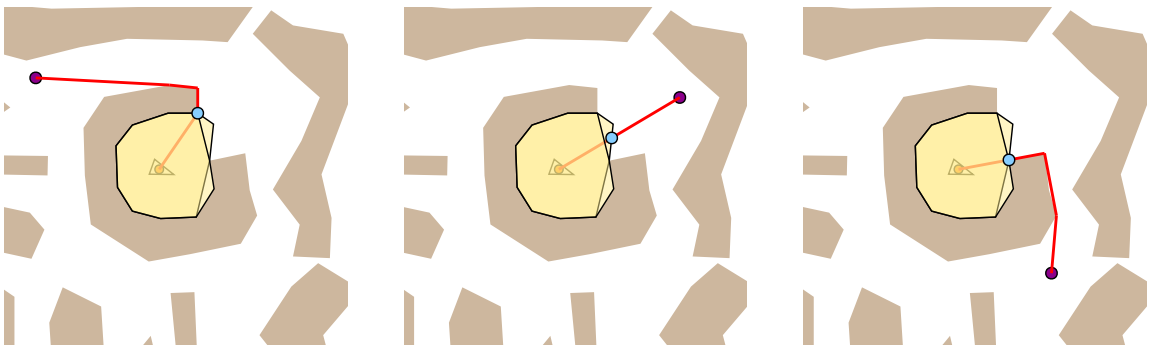
Algorithm 10: WRP Adaptation Procedure

Input: $\mathcal{T} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{T})$ - triangular mesh of $\mathcal{W}$
Input: $\boldsymbol{P}$ - set of convex polygons associated to triangles (convex cover of $\mathcal{W}$)
Input: (k, G, μ, α) - parameters of SOM
Output: $(\nu_1, \ldots, \nu_M)$ - nodes representing a route

```
r ← initialization                                   // creation of ring of nodes
repeat
    I ← ∅                                            // set of inhibited nodes
    T_c ← triangles covered by the current ring r    // ring coverage
    foreach T ∈ Π(T) do        // select T from a random permutation of T
        if T ∉ T_c then
            p_a ← centroid(T)                        // attraction point
            ν* ← select winner node to p_a, ν* ∉ I   // approx. the shortest path
            P_c ← {all associated convex polygons to T}
            if ν* ∉ P, P ∈ P_c then
                adapt(ν*, p_a)
            T_c ← T_c ∪ {T|T ∈ P, P ∈ P_c}
            I ← I ∪ {ν*}                             // inhibit the winner node
        adaptation to triangles
    G ← (1 − α) · G                                  // decrease the gain
until all triangles are covered by the current ring
```

in some polygon[3] of $\boldsymbol{P}_c$.

To avoid placement of nodes unnecessary close to the attraction point p_a, the adaptation rule adapt is modified to find an alternate point for the adaptation. Assume a winner node ν that is being adapted to the centroid p_a of the triangle T and let T has associated convex polygons $\boldsymbol{P}_c$. An approximation of the shortest path from ν to p_a is a sequence of points $(v_1, \ldots, v_k)$, where $v_1 = \nu$ and $v_k = p_a$. The alternate point is found as the farthest intersection point of the segment (v_{k-1}, v_k) with $P \in \boldsymbol{P}_c$ from p_a, see Figure 7.5.

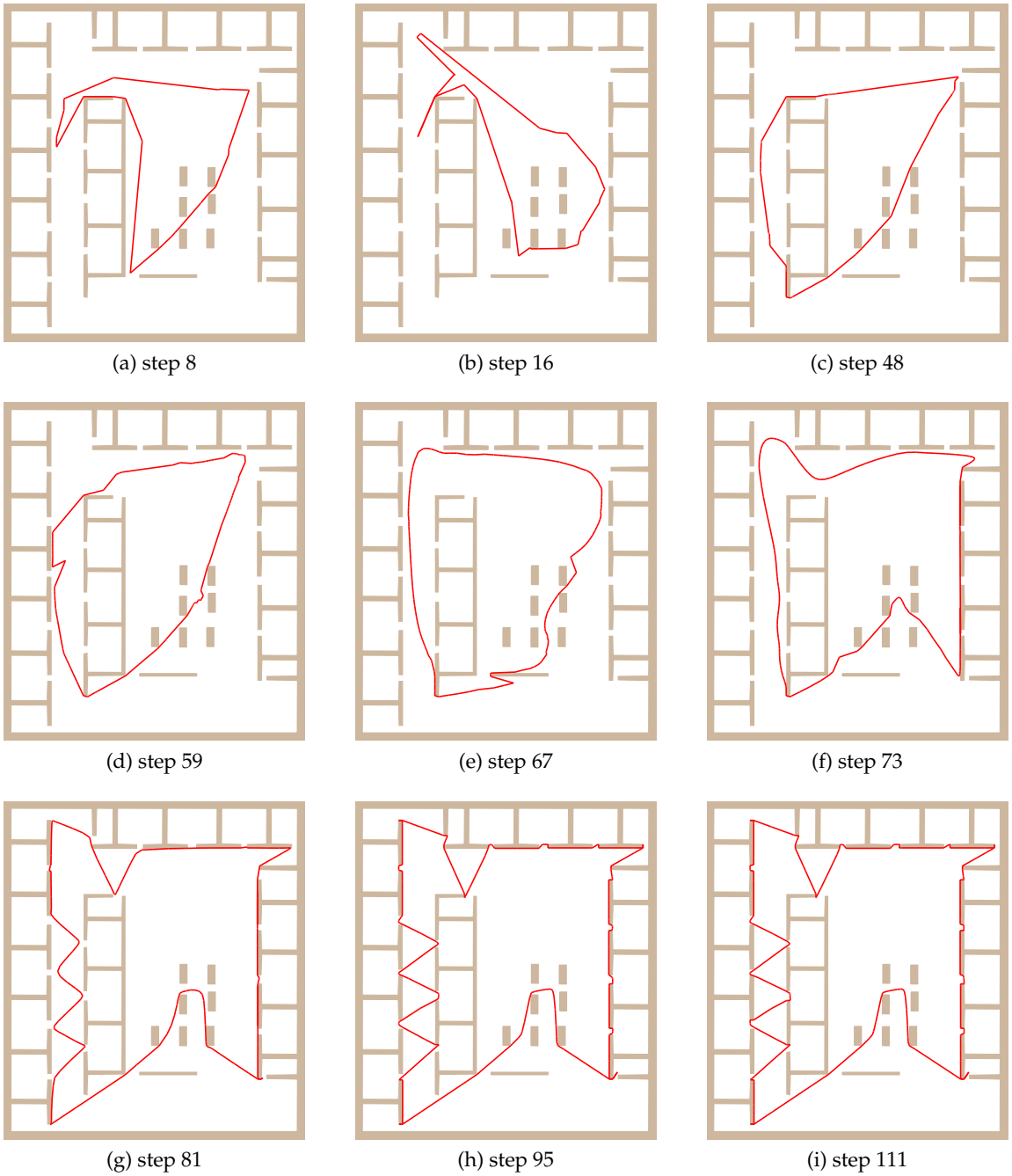


(a) (b) (c)

Figure 7.5: An example of alternate points for a different node and the same attraction point, the attraction point is the centroid of the small triangle inside associated convex polygons, red line segments represent an approximation of the shortest path.

[3] A node can be in such polygon due to its movement during adaptation to another triangle.

An example of the algorithm performance is shown in Figure 7.6. In last thirty steps, a shape of the ring is almost same and winner nodes are moved towards uncovered parts of $\mathcal{W}$, while coverage of the ring is preserved.



(a) step 8 (b) step 16 (c) step 48

(d) step 59 (e) step 67 (f) step 73

(g) step 81 (h) step 95 (i) step 111

Figure 7.6: Performance of the SOM algorithm for the WRP, map *jh*, triangular mesh with 1417 triangles, 100 convex polygons of convex cover for the unrestricted visibility range.

An important aspect of the used triangular mesh should be noted. The number of triangles can be relatively high. For example the typical number of triangles is around one thousand for high visibility range and several thousands for visibility range one meter for examined environments presented in the previous chapters. The high number of tri-

angles requires more computational time to select winner nodes, especially for the first adaptation steps. Two times more nodes than cities is recommended in [242], which is needlessly high for the WRP adaptation, because many triangles are covered from the ring and they are never really presented to the network. The number of nodes can be lower and should correlate with the number of convex polygons rather than to the number of attraction points (centroids of triangles). For high visibility range unnecessary nodes only increase computation burden, because winner nodes are selected from larger set. After several adaptation steps, the ring covers majority of $\mathcal{W}$, thus a winner node is determined only for few uncovered triangles. It means the winner selection is less computationally demanding. The quality of solution is not affected if the ring contains sufficient number of nodes.

Also another aspect of the WRP algorithm with relation to the TSP variant should be noted. It is not necessary to precompute the visibility graph for cities, because alternate points are computed during node adaptation. Instead of the approximate shortest path between a node and the city, the approximation of shortest path between two arbitrary points described in Section 5.2.4 is used, hence only shortest paths between map vertices are utilized.

7.3 Multiple Watchmen Route Problem - MWRP

An extension of the previous algorithm for the WRP to solve the MWRP is similar to extension of the TSP to the MTSP. The main difference is that the MWRP formulation does not consider a common depot, a solution of the MWRP consists of a set of independent patrolling routes.

The flexibility of the SOM approach allows extension to address both MWRP variants: with and without the common depot. The adaptation procedure for both variants is depicted in Algorithm 11. The MinMax criterion is considered in the same manner like in the MTSP, particularly by weighting of distance between the node and the attraction point to prefer selection of a node from shorter rings. However the winner node is selected according to the attraction point, the node can be then adapted to the alternate point like in the WRP approach.

7.3.1 MWRP - Independent Patrolling Routes

The algorithm for the problem variant without the common depot is almost identical to Algorithm 10, the only difference is maintenance of m rings for m watchmen and determination of ring lengths to preferred selection of nodes from shorter rings. The first *foreach* loop of Algorithm 11 is not considered in this MWRP variant.

An example of the algorithm performance is depicted in Figure 7.7. It is shown that rings are separated in the first steps, and rings are expanded after 50 adaptation steps. The solution is almost found in the step 67, but additional 29 steps are necessary to complete coverage. The entry to the room is filed in same cases, because in final stages several attraction points from the same room are presented to the network. This imperfection does not affect the separation of the routes as it is negligible in comparison to a length of the route.

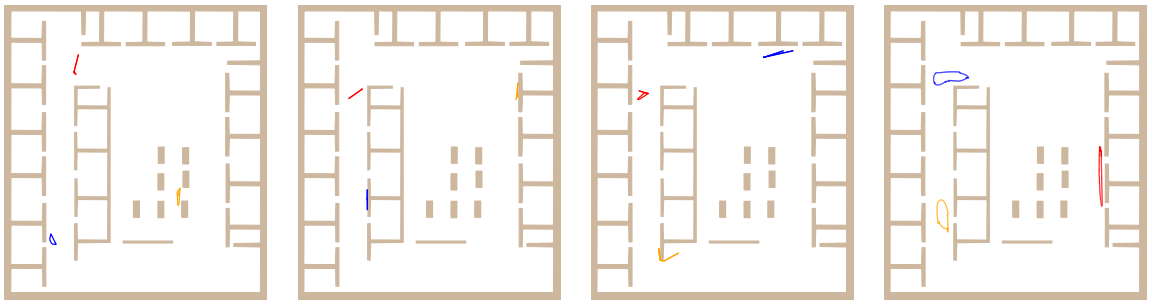
Algorithm 11: MWRP Adaptation Procedure

Input: $\mathcal{T} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{T})$ - triangular mesh of $\mathcal{W}$
Input: $\boldsymbol{P}$ - set of convex polygons associated to triangles (convex cover of $\mathcal{W}$)
Input: m - number of watchmen
Input: c_d - common depot
Input: (d, G, μ, α) - parameters of SOM
Input: δ - maximal allowable distance to c_d
Output: $\{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_m\}$ - routes represented by rings

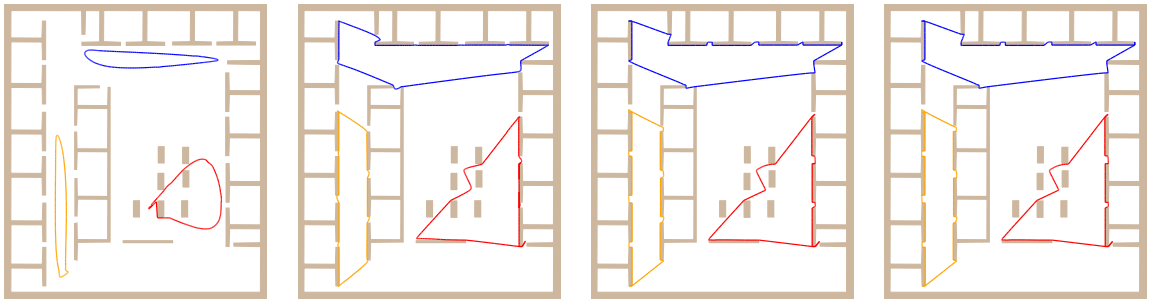
```
initialization                                   // create supporting structures
R ← {r_1, ..., r_m}                                              // create rings
repeat
    I ← ∅                                          // set of inhibited nodes
    error ← 0
    T_c ← all triangles covered by the current rings R
    foreach r ∈ R do                             // adapt each ring to the depot
        ν* ← select winner node from r to c_d, ν* ∉ I
        adapt(ν*, c_d)                                 // adaptation to the depot
        error ← max{error, |ν*, c_d|}
        I ← I ∪ {ν*}                                // inhibit the winner node
        adaptation to the depot - only for the depot variant
    foreach T ∈ Π(T) do      // Π(T) is a random permutation of triangles
        if T ∉ T_c then                          // T is not already covered
            a ← centroid(T)                                 // attraction point
            ν* ← select winner node to a, ν* ∉ I
            P_c ← {all associated convex polygons to T}
            if ν* ∉ P, P ∈ P_c then
                adapt(ν*, a)
            T_c ← T_c ∪ {T|T ∈ P, P ∈ P_c}
            I ← I ∪ {ν*}                            // inhibit the winner node
        adaptation to triangles
until (all triangles are covered) ∧ (error < δ)
```

7.3.2 MWRP with the Common Depot

To attract rings to the common depot a winner node from each ring is selected and moved towards the depot. The alternate point is not determined in the adaptation procedure for the depot, because the depot is a point, which have to be visited, therefore the $error$ variable is maintained like in the MTSP. An example of the algorithm performance is shown in Figure 7.8. Found routes are of course, due to the depot, longer than in the case of the MWRP without a depot.

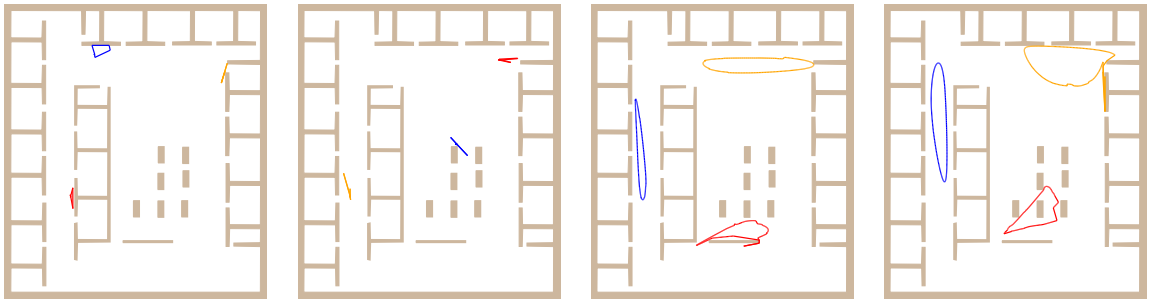


(a) step 8 (b) step 14 (c) step 31 (d) step 50

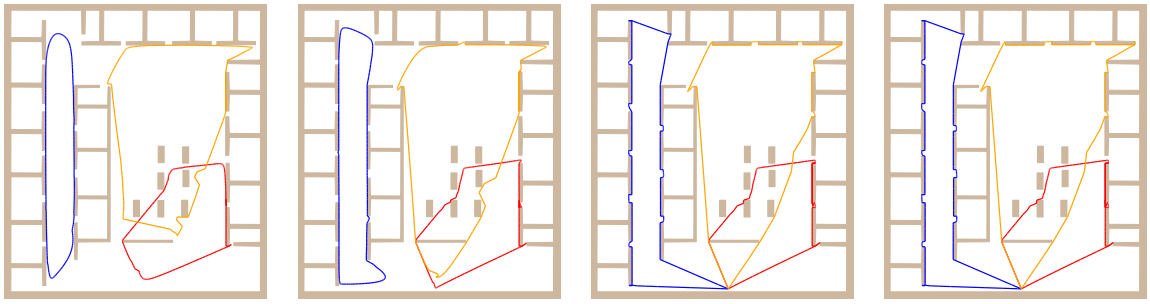


(e) step 54 (f) step 67 (g) step 82 (h) step 96

Figure 7.7: Performance of the SOM algorithm to solve the MWRP without a common depot, map *jh*, triangular mesh with 1417 triangles, 100 convex polygons for unrestricted visibility range, lengths of found routes are 37, 44 and 35 meters.



(a) step 16 (b) step 30 (c) step 52 (d) step 54



(e) step 62 (f) step 65 (g) step 84 (h) step 101

Figure 7.8: Performance of the SOM algorithm to solve the MWRP with the common depot, map *jh*, triangular mesh with 1417 triangles, 100 convex polygons for unrestricted visibility range, lengths of found routes are 43, 62 and 58 meters.

7.4 Experiments

The proposed WRP adaptation procedure has been experimentally verified in a set of environments represented as polygonal maps and for a set of selected visibility ranges. Particular maps properties are depicted in Table 7.1, where the last column denotes the number of convex polygons of the convex polygon partition used for computation of the approximate shortest path. A convex polygon partition is found by Seidel's algorithm [233]. The first four environments are maps of real buildings.

Map	Size [m × m]	No. Vertices	No. Holes	No. Convex Polygons
jh	20.6 × 23.2	196	9	77
pb	133.3 × 104.8	137	3	50
ta	39.7 × 46.8	101	2	46
h2	84.9 × 49.7	1061	34	476
dense	21.0 × 21.5	288	32	150
potholes	20.0 × 20.0	153	23	75
warehouse	40.0 × 40.0	142	24	83

Table 7.1: Properties of environments, the first four maps represent real buildings.

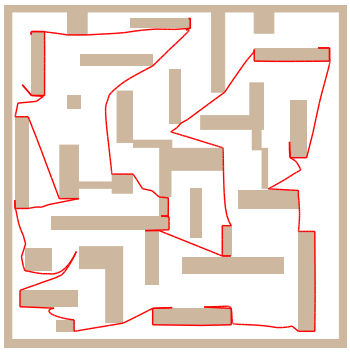
For each environment and visibility range d a triangular mesh is created by the quality mesh generator [237] for the minimal required angle 32.5° and 25.0° for the map *jh*, and the maximum triangle area. The area has been set experimentally according to the circumscribed circle of the triangle, which radius is derived from d.

The first three maps (*jh*, *ta* and *pb*) are used in comparison of the proposed WRP algorithm with a reference solution found by the decoupled approach as a solution of the AGP and consecutive the TSP. The AGP is solved by the algorithm CPP, Section 4.3.1. The CPP algorithm has been selected mainly due to its similarities with the supporting convex cover of the proposed WRP algorithm. The TSP is solved exactly by the Concorde solver [16] and by the proposed SOM algorithm with the approximate shortest path, Section 5.2.2, particularly with the full path refinement (*pa*) variant and the *error* stop condition. The same parameters from Table 5.2 are used for both SOM algorithms: TSP and WRP. The only exception is the number of neurons, which has been set individually for each problem in case of the proposed WRP algorithm. The supporting approximation of the shortest path in the WRP algorithm is the two points variant with the full path refinement, Section 5.2.4.

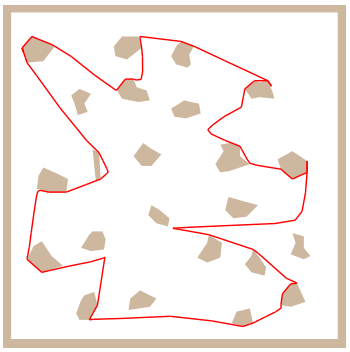
Both the SOM algorithms are randomized, therefore each particular problem is solved 20 times and their performance is compared by the percentage deviation to the reference path length of the mean solution value, $PDM = (\overline{L} - L_{ref})/L_{ref} \cdot 100\%$, and the percentage deviation from the reference of the best solution value (PDB), where L_{ref} is the length of the path found by Concorde.

Experimental results are presented in Table C.1. The best solutions of selected problems found by both approaches (WRP and AGP+TSP) are presented in Figure C.2 and several additional solutions of the WRP in Figure 7.9.

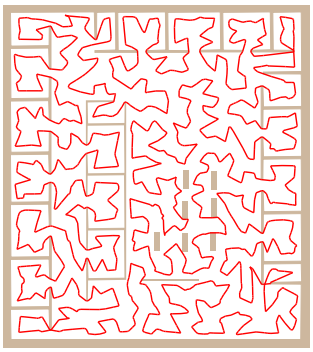
However found solutions are not compared with exact solutions, regarding the figures one can expect that solutions (for high visibility ranges) are very close to optimum.



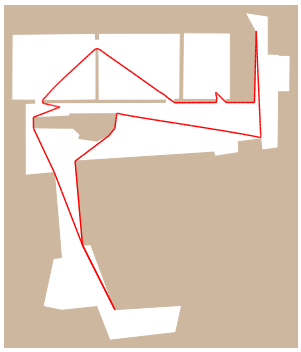
(a) map *warehouse*, d = 10 m



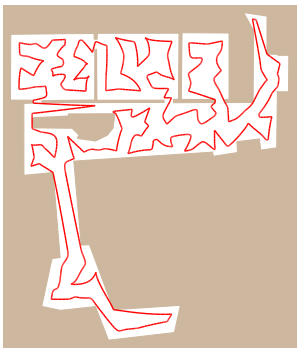
(b) map *potholes*, d = 10 m



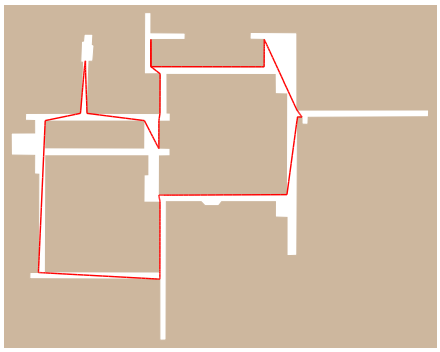
(c) map *jh*, d = 1 m



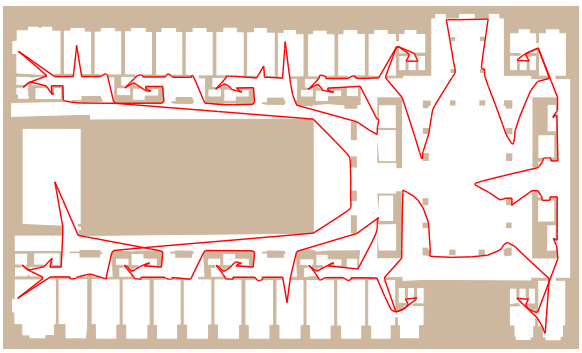
(d) map *ta*, d = 10 m



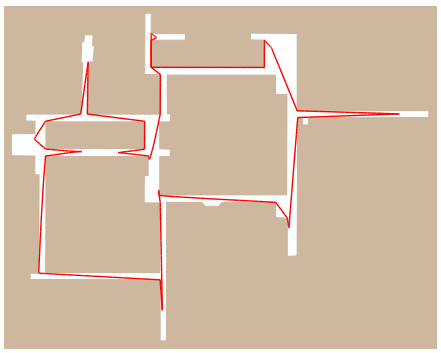
(e) map *ta*, d = 2 m



(f) map *pb*, d = *inf*



(g) map *h2*, d = 10 m



(h) map *pb*, d = 10 m

Figure 7.9: WRP, selected solutions for various visibility range d.

Lengths of found routes by the proposed WRP algorithm are always shorter than solutions of the TSP found by the SOM adaptation procedure. If the *va-1* path refinement variant of the SOM adaptation procedure for the TSP is used, solutions are found about twenty percents faster and lengths of found paths are longer about 0.5% at maximum.

The required number of adaptation steps of the WRP algorithm is more than one hundred and for the map *pb* and d=1 m the algorithm has been terminated after 180 steps. It means that the found path does not provide the full coverage, the over all coverage is more than 99.9% and in the worst case it is higher than 98.4%. The convergency issue can be caused by two factors. At first, the number of used neurons is low. The second is related to the SOM parameters used that are probably not well suited for problems with high number of triangles. For the *pb* problem with d=1 m it means that 14 462 triangles have to be covered.

Required Computational Time

All algorithms have been implemented in C++ and compiled by the G++ 4.2 compiler with -O2 optimization flags. All experiments have been performed within the same computational environment using single core of the Athlon X2 5050e at 2.6 GHz CPU, 2 GB RAM running FreeBSD 7.1. The required computational time the algorithms depends on the number of neurons, which is related to the number of triangles (WRP) and guards (AGP+TSP), therefore average values of T for the selected numbers of neurons are presented in Figure 7.10

The presented times do not include computation of all supporting structures. Construction of the convex cover from the triangular mesh, takes a fraction of second for high visibility ranges and less than two seconds for a triangular mesh with seven thousands triangles. A triangular mesh and a convex polygon partition is found in less than one hundred milliseconds, also supporting visibility graphs are found in a fraction of second. Regarding the time to solve the WRP or the TSP, required computational time to create these structures is negligible. The most time consuming preparation step is computation of all shortest path between map vertices, the required time is included in the presented results, and also it is included in T in Table C.1.

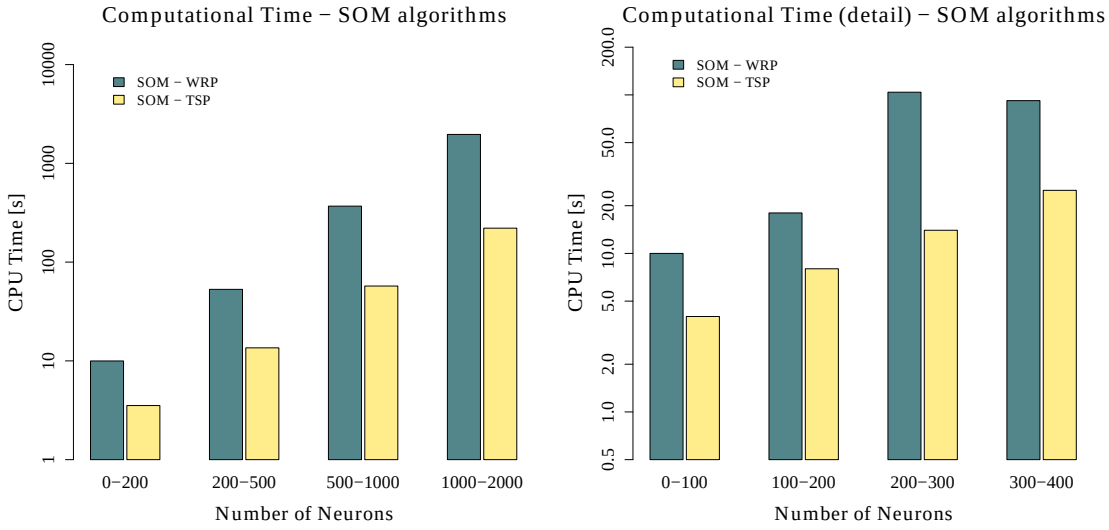


Figure 7.10: Required computational time of the SOM algorithms.

MWRP variants

An application of the proposed adaptation procedure to address the MWRP variant with a common depot is demonstrated by several examples of found solutions in Figure C.3. The performance is not explicitly compared with a reference method. The expected performance of the algorithm should be similar to the WRP and the MTSP algorithm, which has been compared with the GENIUS heuristic in Chapter 5.

If the MWRP is solved without the common depot, independent patrolling routes are found, see Figure C.4.

7.5 Discussion

A new algorithm to find an approximate solution of the WRP and MWRP with restricted visibility range in a polygonal domain have been proposed in this chapter. An algorithm for the MWRP with d-visibility in a polygonal domain has not been found in literature, thus the proposed algorithm is probably the first soft-computing algorithm for this kind of problems.

The proposed SOM procedure is based on the geometrical interpretation of neurons weights that are considered as nodes in a polygonal domain. The interpretation is the main motivation for applied supporting geometrical structures, which are essential for a reasonable computational requirements. The ring of nodes represents the watchman route that evolves in a polygonal domain, the nodes are moved towards to the attraction point. Four main issues are addressed by the supporting structures: selection of the attraction point, a determination of a path in order to select a winner node, a node movement towards the attraction point, and a computation of the ring coverage.

Despite the fact that the SOM procedure is relatively simple, the proposed adaptation procedure for a polygonal domain $\mathcal{W}$ becomes relatively complex, because of structures and algorithms to support path and visibility queries. The supporting structures and algorithms are summarized in the following list:

- a convex partition of $\mathcal{W}$,
- a convex cover of $\mathcal{W}$,
- a triangular mesh of $\mathcal{W}$,
- the visibility graph,
- all the shortest paths between vertices of $\mathcal{W}$,
- a point-location algorithm,
- an approximation of the shortest path,
- the straight walking algorithm in a triangular mesh and a convex partition.

On the other hand, the advantage of these structures and algorithms is their simplicity and computational feasibility.

The proposed adaptation procedure has been experimentally verified in several environments, which represent real indoor environments, and compared with the decoupled approach based on solutions of the AGP and the related TSP. According to the experimental results, the proposed algorithm provides better solutions (from the length of the tour point of view) than the decoupled approach based on the CPP algorithm and the SOM algorithm for the TSP. Regarding the results presented in Chapter 4 the quality of found solutions by the proposed BP algorithm is competitive with the proposed direct solution of the WRP.

However the algorithm is able to find better solutions than the decoupled approach, it can be improved in two aspects. At first, the performance of the algorithm is relatively poor for small visibility ranges, especially in comparison to outstanding results for high visibility ranges. The worse performance can be caused by the used convex cover, which does not provide good alternate points. Moreover it can also be caused by a node movement towards another alternate point. The attraction points are presented in a random order. The presentation can be eventually modified to consider the triangle coverage, to prefer triangles with small coverage, see Figure 7.4. Such triangles have to be covered explicitly, because they cannot be covered from paths to other parts of the environment.

The second aspect relates to the used SOM procedure [243] and used parameters, Table 5.2. Different values of parameters or modification of the neighbouring function can decrease the required number of adaptation steps and possibly improve the convergency. The adaptation procedure can be more sophisticated, e.g. it can use the local search strategy to select the winner node, proposed in the Co-Adaptive net algorithm [56], to avoid unnecessary computation of the node–point distances. Also the used number of neurons is fixed for the whole adaptation. Higher number of neurons is not necessary after rings are expanded, because larger portion of the free space is covered by the ring and not by the nodes. A consideration of dynamic number of neurons, like in [214] can be helpful. Besides, so-called competition strategy described in [214], can be considered in the proposed WRP algorithm. The strategy uses the nearest edge of the ring instead of the nearest node in the competitive rule, thus it can be useful to attract the ring into parts that are close to the watchman route, but nodes of the ring are relatively far.

In addition to proposed improvements, one of the future work is a comparison of the proposed (or improved) SOM algorithm with the heuristics approach [211], which is similar in an aspect of route reduction. It uses an initial AGP solution and reduces the length by moving the guards, while the proposed WRP algorithm uses attraction points and finds alternate points. The main difference of the proposed algorithm is that instead of sophisticated heuristics for the AGP, a simple underlying structure (a triangular mesh) is used. Also the proposed SOM procedure provides flexibility to addressed the restricted visibility range and variants of the WRP.

With respect to the quality of found solutions for small values of d it should be mention that for small visibility distances the *d-sweeper route problem* is very close to the coverage task by a mobile robot, which is solved by algorithms based on a cell decomposition and explicit routing shapes in the cells [11]. These algorithms seem to provide more suitable solutions for a real robot than an approximate solution of the TSP.

The MWRP with and without a common depot have been solved for various number of watchmen in different environments. The most interesting aspect of the SOM based MWRP algorithm in relation to the path planning is non-crossing paths. Similarly to the MTSP the SOM adaptation procedure provides non-crossing path, however with lower frequency than in the MTSP.

The proposed algorithm addresses variant of the WRP that can be found as *d-sweeper route problem* in the literature. An application of the algorithm to address the so-called *d-watchman route problem* is straightforward, only triangles of convex polygons that are connected with the border of $\mathcal{W}$ have to be considered.

Despite of mentioned imperfections, the proposed approach combines the self-organizing principle and supporting geometrical structures to address problems studied in the computational geometry, thus it can be probably applied for other problems from the class of hybrid visibility problems. The most interesting problem is the *vision points problem* or the VPP that aims to combine the sensing and motion costs.

Chapter 8

Multi-Goal Path Planning with Trajectory Generation

The SOM adaptation procedures for the multi-goal path planning problem have been presented in the previous chapters. Moreover the variants with the MinMax criterion have been considered for a group of cooperating mobile robots. At first, the shortest-path roadmap approach has been applied for the TSP problem formulation, then navigation functions based on the APF have been used. These approaches satisfy kinematic constraints for a robot with differential nonholonomic drive and additional motion constraints can be satisfied if a trajectory is considered. Velocity or acceleration limits constraint the velocity profile that can provide more accurate estimation of the required time to travel. The velocity profile can be used to consider cost of the path with respect to additional criterions[1]. In addition, a trajectory can be used to guarantee coordination of motion of several mobile robots. The crossing paths do not necessary mean collision, but the coordination cannot be guaranteed without known position of the robot in time.

Trajectories can provide more accurate plans, however the problems of trajectory generation and trajectory following can be considered as a part of the control theory, where problems are studied as state models and the optimal control is one of the desired goals. On the other hand, problems of the multi-goal path planning considering obstacles and cooperating mobile robots studied in this thesis are related to the Artificial Intelligence domain. Although these two domains are quite different, nowadays techniques of the motion planning try to find trajectories that can be directly used for controlling the robot. The RRT technique is an example of such approach, which is also able to find trajectories for a robot with complex kinematics or for several robots.

This chapter is dedicated to consideration of the trajectory generation during the SOM adaptation. The examined problem is the MWRP, because in this problem desired goal locations are not prescribed, hence trajectory can freely evolved in the free space. The main motivation of the studied problem of trajectory generation in the multi-goal path planning problem is to find more accurate plan. Besides, such technique can provide paths for complex robots with multiple bodies and additional motion constraints.

The rest of the chapter is organized as follows. The next section describes preliminary results of consideration of velocity profiles during the multi-goal path planning. The results are discussed in Section 8.1.3 where an idea of combination of the SOM adaptation

[1] For example in [149] authors reported up to 10 % energy savings by their trajectory generation compared with the energy optimal trapezoidal velocity profile and loss-minimization control.

procedure with trajectory generation is proposed. Section 8.2 is dedicated to the motion planning where a new multi-goal motion planning algorithm called RRT–Path^{ext} is proposed. The adaptation procedure with a ring regeneration based on the RRT–Path^{ext} is described in Section 8.3, the procedure is applied to the WRP with trajectory generation. The last section of the chapter is dedicated to discussion of future work.

8.1 Preliminary Work

Two types of velocities profiles have been considered in the preliminary study of the multi-goal path planning. The first type of the profile is derived from the path curvature, while the second type is based on the time optimal trapezoidal velocity control. Computation of the curvature requires first and second derivatives, therefore paths have to be smooth. The trapezoidal profile is suitable for the turn-move motion along straight line segments. In the next sub-sections, two methods to find velocity profiles for given paths are described. Both methods can be considered as time optimal, because they maximize forward velocity of the robot along the path.

8.1.1 Velocity Profile for Smooth Paths

The first approach considering the acceleration limits during the SOM adaptation has been proposed in [86]. It is based on spline curves describing the robot trajectory. The trajectory is formed from two splines $x(u)$ and $y(u)$, where u is the parameter along the curve. Each spline consists of several segments - cubic polynomials. Knots are tangent points of two neighbouring segments with continuous derivatives. A path found as a ring of nodes is used to define a curve. The spline curve is determined by the start point of the path, the end point of the path, their derivatives, and knots. Knots are selected to fit the points of the path, except the start and the end point of the path. To fully define the spline, additional points are determined as an average point between two points of the path.

A velocity profile along the trajectory can be computed from the curvature and acceleration limits. The curvature is defined as

$$\kappa(u) = \frac{x'(u)y''(u)'' - x''(u)y'(u)}{\sqrt[3]{(x'(u)^2 + y'(u)^2)^2}}. \tag{8.1}$$

The maximal forward velocity v can be computed from $v = \omega_{max}/\kappa$, where ω_{max} is the maximal radial speed. Considering these relations the velocity profile can be computed in following steps.

1. Local extremes of the curvature are determined and denoted as $\{p_1, \ldots, p_n\}$. The robot has to moved with allowed maximal speed due to restricted radial velocity in these points. The curve radius is bigger before and after these points, therefore the robot can move faster.
2. For each p_i the maximal velocity profile is determined. The profile has shape of 'U' (or 'V'), because the robot tangentially decelerates before p_i and accelerates after p_i with respect to the acceleration limits.
3. Velocity profiles for the initial and final points of the path are obtained in similar manner.
4. The highest allowable overall velocity profile is determined as the minimum of all profiles.

An example of such velocity profile is shown in Figure 8.1.

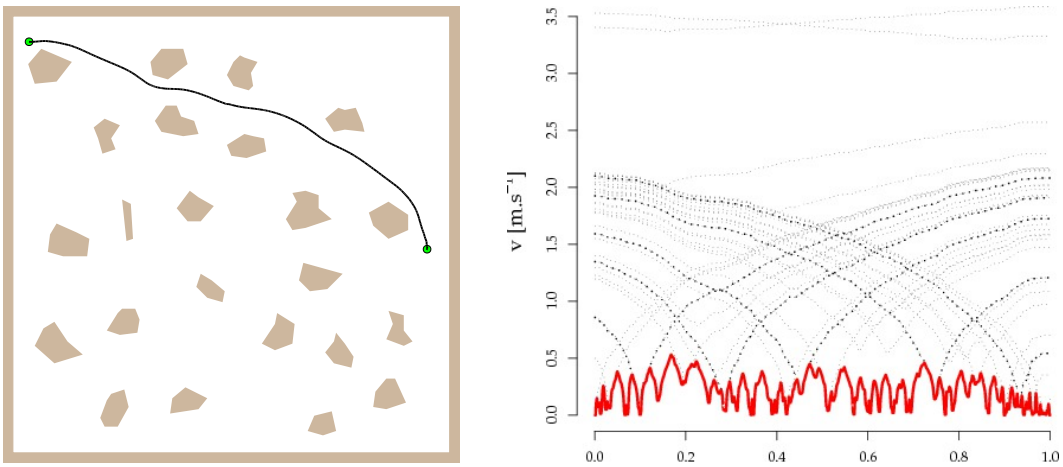


Figure 8.1: A path and velocity profiles along the path (parameter u).

If it is necessary to consider the tangential and the radial accelerations, e.g. to consider limited friction force and to satisfy the pure rolling condition of the robot wheels, the overall acceleration can be bounded by $a = \sqrt{a_{tang}^2 + a_{rad}^2}$, where a_{tang} is the tangential acceleration and a_{rad} is the radial acceleration. The forward velocity v can be then determined from $a_{rad} = v^2\kappa$ and the above described procedure can be used to find highest allowable profile. Consideration of the friction force is necessary for very fast robots, where limits are not in motors or mass of the robot, e.g. robots in the robotic soccer [87].

The presented approach applied to paths composed from straight line segments suffers by the following issue. The path is transformed into spline curves $x(u)$, $y(u)$, which are smooth, but they can collide with obstacles. Smooth navigation functions may be used, e.g. the harmonic potential field functions presented in Section 6.1.2, to avoid the issue.

8.1.2 Velocity Profile for Straight Line Segment Paths

If a robot is able to turn at a place, e.g. it has differential nonholonomic drive, the most straightforward control strategy is following the line segment and turning at the end of the segment in the direction of the next segment. Such motion strategy can be called *turn-move* and it is well suited for the shortest-path roadmap based on the visibility graph. The velocity profile for one segment has trapezoidal shape. The robot has zero forward velocity at the beginning of the segment, then the velocity is increased up to the maximal speed of the robot, while the maximal allowable acceleration is considered. The robot decelerates before the end of the segment and it stops at the end. The robot turns with similar profile for the radial velocity, however acceleration and deceleration phases are negligible for robots like the P3AT [3].

The TSP with the cost of a path between cities computed as the travel time has been studied in [263]. The travel time has been computed from the trapezoidal velocity profiles. The GENI heuristics algorithm for the TSP has been modified to consider a turn angle at cities. The modification is necessary, because the cost of the route is not affected only by costs between two cities, but the angle depends on the previous and the next city of the planned route.

Required times to travel found by the modified algorithm has been compared with the times to travel for the route found by the original algorithm. The comparison of these times has been performed for various problems and several maximal radial accelerations. The robot has been modeled with following parameters: the maximal forward speed 1.5 m.s^{-1}, the maximal radial velocity 6 rad.s^{-1}, the maximal forward acceleration and deceleration 1 m.s^{-2}, the maximal radial acceleration in the range from 0.1 to 7 rad.s^{-2}. Experimental results are shown in Figure 8.2 as dependency of the time ratio ϑ on the radial acceleration ψ. Four problems have been examined in the comparison, the number of cities is denoted as n. The results indicate that for small problems and high radial

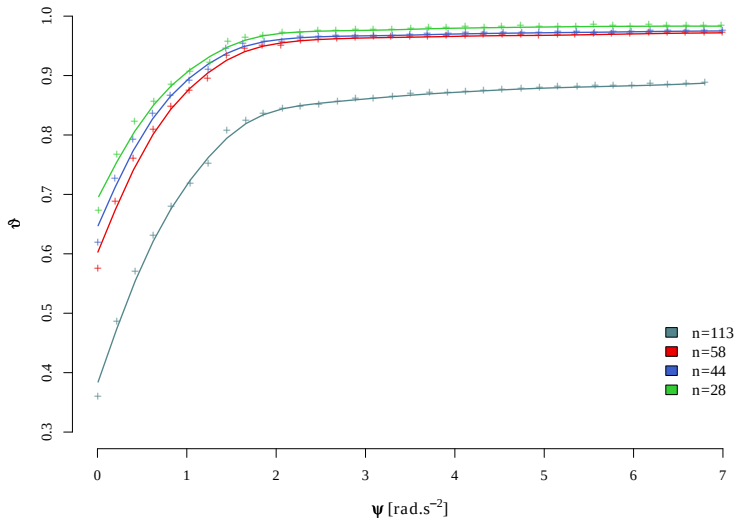


Figure 8.2: Comparison of TSP solutions according to maximal allowable radial acceleration.

acceleration it is not necessary to consider angles and velocity profiles during the tour construction and its improvement, the quality of the solution is about one percent better if velocity profiles are considered. For the problem with 113 cities the solution found by modified algorithm is more than about ten percents better.

Similar comparison has been presented in [185], where paths from the shortest-path roadmap and from the APF approach have been compared. The time to traverse the complete tour has been computed for the turn-move strategy in the case of the roadmap, while velocity profiles for smooth paths, Section 8.1.1, has been used for the APF. However the smooth paths are longer than the sequences of the straight line segments, the overall time to travel the tour is about three percents lower.

8.1.3 Discussion of Preliminary Work

The preliminary experiments show possible improvement of quality of solution if a trajectory is considered during the multi-goal planning. The more important aspect of considering trajectory is a planning of a feasible path for a robot with different kinematics, e.g. car-like robot or complex robots like [148]. The successful application of navigation functions in the SOM adaptation procedure described in Chapter 6 motives to combine the adaptation procedure with a motion planner to satisfy various kinematic or kinodynamic constraints.

The SOM adaptation procedure can be used for high dimensional input, so it can be possibly used for high dimensional configuration space where nodes can represent points in $\mathcal{C}$, or more precisely points in $\mathcal{C}_{free}$. The main issue of such approach relies on the efficient determination of the distance between a node and a goal, which is exactly the same problem in any randomized sampling based motion planner for high dimensions. Due to this difficulty the first step of the multi-goal motion planning with SOM is based on the following ideas of combination of the trajectory generation and the route optimization.

- The SOM adaptation procedure can be used as the route optimization procedure.
- The ring of nodes can be considered as a sequence of points in $\mathbb{R}^2$, the ring (path) is two dimensional projection of the trajectory from the high dimensional $\mathcal{C}$.
- The SOM approach provides an intuitive way (simple competition rule) to consider the MinMax criterion in the multi-goal planning for several robots. The cooperative motion can be addressed by the SOM competitive rule, which prefers nodes from shorter rings.
- The ring can be used to determine a sequence of visits of goals.
- A trajectory to visit the found sequence of goals can be found by a motion planner.

It is necessary to consider the multi-goal problem in order to select a suitable motion planning algorithm to find trajectories. Mainly because the classical motion planning problem deals only with the problem of finding a trajectory between two points. The motivation is also to consider various motion constraints. Regarding the review of the state of the art the RRT algorithm seems be the most suitable approach. Moreover the RRT–Path algorithm introduced in [264] by Vonásek and Faigl uses an auxiliary path to guide the randomization process of the selection of new configuration, therefore a ring of nodes can be used as the auxiliary path. The RRT–Path is described in the next section, where a new algorithm called RRT–Path^{ext} is proposed to solved the multi-goal motion planning problem. The algorithm is then used in the SOM adaptation procedure to find a solution as a trajectory in the WRP, Section 8.3.

8.2 RRT–Path^{ext} Algorithm - Multi-Goal Motion Planner

8.2.1 RRT–Path Algorithm

One of the main issues of RRT algorithms is the narrow passage problem. The RRT–Path algorithm tries to avoid the problem by a guided sampling of new configurations in $\mathcal{C}$ along an auxiliary path, it has been proposed by Vonásek in [263]. The algorithm is principally same as the original RRT algorithm, but the performance is improved by the guided sampling. For a given initial and desired configurations an auxiliary path has to be found. Let the path is given as a sequence of points $(p_1, \ldots, p_n)$, where the first point is close to the starting position of the robot and the last point is close to the goal. A schema of the RRT–Path is depicted in Algorithm 12.

The algorithm performs in K steps at maximum. A random configuration q_{rand} is sampled and the nearest configuration q_{near} of the tree is found in the each step. According to the model of the robot motion the most suitable control input is selected from the set of possible control inputs $\mathcal{U}$ to extend the tree from q_{near} towards q_{rand}. If the tree can be extend (q_{new} is in $\mathcal{C}_{free}$) and a trajectory from q_{near} to q_{new} is feasible (the robot movements satisfies required constraints) q_{new} is added to the tree and q_{near} is marked as the parent

Algorithm 12: RRT–Path

Input: $\mathcal{C}$ - configuration space of the robot
Input: q_{start}, q_{goal} - initial and desired configurations
Input: $\boldsymbol{P} = \{p_1, \ldots, p_n\}$ - an auxiliary path, where $p_i \in \mathbb{R}^2$
Input: K - maximal number of iterations
Input: g - temporal goal bias
Input: $\mathcal{U}$ - set of possible inputs of robot control
Input: δ - minimal distance of the found trajectory to the goal

```
i ← 0
T.add(q_start)                          // add initial configuration to the tree
q_t ← p_1      // use first point of the auxiliary path as temporal goal
while i < K do
    if temporal goal is reached then
        P ← P \ {q_t}
        q_t ← next(P)                              // select next temporal goal
    if i mod g = 0 then                              // every g-th step used q_t
        q_rand ← q_t
    else
        q_rand ← select random configuration in C
    q_near ← select the nearest neighbour configuration from the tree T to q_rand
    q_new ← extend q_near towards q_rand
    if q_new can be connected to q_near then
        if ρ(q_new, q_goal) ≤ δ then
            break                          // terminate tree growing procedure
        T.add(q_new)
    i ← i + 1                                          // increase step counter
```

of q_{new}. The algorithm is terminated if a configuration of the tree is at sufficient distance to the goal, less than δ.

A point of the auxiliary path (called temporal goal) is used instead of random configuration every g iterations, this technique is similar to the goal bias variant of the RRT. The main difference is in the moving of the temporal goal q_t along the auxiliary path $\boldsymbol{P}$. A robot workspace is considered as planar environment, and even for the high dimensional $\mathcal{C}$ (e.g. robot position, orientation and its velocities or accelerations) the auxiliary path is two dimensional. Therefore q_t uses only two coordinations (x, y) of the point at the auxiliary path. If the tree contains a configuration sufficiently close to a point p from $\boldsymbol{P}$, p is removed from $\boldsymbol{P}$. The auxiliary path is used as a guideline of the tree growing process and it represents knowledge about the environment.

The RRT–Path has been compared with three RRT algorithms in [264], namely with the original version of RRT, RRT-Bidirect and RRT–Blossom. The RRT–Path algorithm significantly outperforms all other algorithms in the length of the found path, size of the tree as well as required computational time. Examples of solutions are shown in Figure 8.3. In addition to the comparison of the RRT algorithm variants, several methods to obtain an auxiliary path have been evaluated. The Segment Voronoi diagram, Visibility-Voronoi diagram [267] and the PRM [146] methods have been used to find an auxiliary path. These methods provide clearance around the path that allows the RRT–Path algorithm to grow around the path. The Visibility-Voronoi diagram with higher clearance provides better re-

(a) original RRT, N=2 542 (b) RRT–Blossom, N=44 573 (c) RRT–Path, N=89

Figure 8.3: RRTs and found trajectories, N is the number of tree vertices.

sults than with lower clearance. The visibility graph has not been used, because it does not provide sufficient clearance and the performance of the RRT–Path algorithm is poor.

However the RRT–Path algorithm provides feasible trajectory from the start to the goal locations (configurations), it cannot be directly used in the multi-goal planning problem. The auxiliary path can be efficiently used only if it is the *monotone polygonal path*. Also consideration of $q_t \in \boldsymbol{P}$ can be too restrictive in cases where obstacle edges are part of the auxiliary path, e.g. shortest-path roadmap. In such cases, q_t can be part of the obstacle edge, which can lead to grow the tree towards the obstacle. Such unsuccessful tree expansion can be repeated several times and a solution is not found in the given maximal number of steps K. These issues are the main motivation of the proposed extension of the algorithm described in the next section.

8.2.2 RRT–Path^{ext} Algorithm

The idea to avoid undesired growing of the RRT–Path algorithm in a direction to obstacles is based on an extension of the auxiliary path to an auxiliary corridor. A corridor supports finding a path with required clearance, which can be more important than length of the path [268]. It is exactly the case of the tree expansion process in the RRT–Path. If q_t is also sampled around the path (in $\mathcal{C}_{free}$) and not only at the path, the tree will more likely expand into the free space rather than in a direction to the obstacle.

A triangular mesh of the free space of the robot workspace $\mathcal{W}$ can be used to find a corridor with defined distance from the auxiliary path. The main advantage of the triangular mesh is simple and feasible determination of the shortest path by Dijkstra's algorithm. The procedure to find an auxiliary path and its corridor in the triangular mesh is depicted in Algorithm 13. The start and goal positions are added to the mesh and the auxiliary path is found as the shortest path between two vertices in the mesh.

An auxiliary path with a corridor is the triplet $(\boldsymbol{V}_{aux}, \boldsymbol{E}_{aux}, \boldsymbol{T}_{aux})$, where $\boldsymbol{V}_{aux}$ is the sequence of vertices representing the shortest path, $\boldsymbol{E}_{aux}$ is the sequence of edges e_i such that $\mathrm{source}(e_i) = v_{i-1}$ and $\mathrm{target}(e_i) = v_i$, and $\boldsymbol{T}_{aux}$ is the set of triangles representing the corridor. Each triangle T from $\boldsymbol{T}_{aux}$ is associated to the closest vertex v of $\boldsymbol{V}_{aux}$, i.e. the particular vertex v has associated a set of such triangles.

An initial auxiliary path can also be found as an approximation of the shortest path by

Algorithm 13: Find an auxiliary path and its corridor

Input: $\mathcal{T} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{T})$ - a triangular mesh of $\mathcal{W}$
Input: p_{start} - a start position
Input: p_{goal} - desired goal position
Input: δ - corridor clearance
Output: $\boldsymbol{P}_{aux} = (\boldsymbol{V}_{aux}, \boldsymbol{E}_{aux}, \boldsymbol{T}_{aux})$ - found auxiliary path with corridor

$\mathcal{T} \leftarrow$ *extend* $\mathcal{T}$ *by points* p_{start} *and* p_{end}
$\boldsymbol{E}_{aux} \leftarrow$*find path from* p_{start} *to* p_{end} *by Dijkstra's algorithm in the graph* $G(\boldsymbol{E}, \boldsymbol{V})$
$\boldsymbol{T}_{aux} \leftarrow \emptyset$
foreach $v \in \boldsymbol{V}_{aux}$ **do**
  $\boldsymbol{T}_{aux} \leftarrow \boldsymbol{T}_{aux} \cup \{T |$*at least one vertex of* T *is closer to* v *than the distance* $\delta\}$

the algorithm proposed in Section 5.2.2 or by any path planning algorithm. If a path is collision free, then the auxiliary corridor for the path can be found by Algorithm 14. The

Algorithm 14: Find an auxiliary corridor

Input: $\mathcal{T} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{T})$ - a triangular mesh of $\mathcal{W}$
Input: P - path as a sequence of points
Input: δ - corridor clearance
Require: P is collision free in $\mathcal{W}$
Output: $\boldsymbol{P}_{aux} = (\boldsymbol{V}_{aux}, \boldsymbol{E}_{aux}, \boldsymbol{T}_{aux})$ - found auxiliary path with corridor

$\mathcal{T} \leftarrow$ *extent* $\mathcal{T}$ *by points* p_{start} *and* p_{end}
$\boldsymbol{T}_{incident} \leftarrow$*find all incident triangles with the path* P `// use walking in trimesh`
$\boldsymbol{V}_{incident} \leftarrow \{v | v \in T \wedge T \in \boldsymbol{T}_{incident}\}$ `// all vertices of triangles` $\boldsymbol{T}_{incident}$
$\boldsymbol{V}_{aux} \leftarrow$*use points of* P *as vertices of the auxiliary path* `// create vertices from` P
$\boldsymbol{E}_{aux} \leftarrow$*set of segments (edges) of the path* P `// create edges from` P
$\boldsymbol{T}_{aux} \leftarrow \emptyset$
foreach $v \in \boldsymbol{V}_{incident}$ **do**
  $\boldsymbol{T}_{aux} \leftarrow \boldsymbol{T}_{aux} \cup \{T |$*at least one vertex of* T *is closer to* v *than the distance* $\delta\}$

algorithm is similar to Algorithm 13, but at first all incident triangles $\boldsymbol{T}_{incident}$ with the path P are found. The path is collision free, therefore all incident triangles are found by the walking in triangular mesh procedure for two consecutive points of the path. After that, points of the path are added to the triangular mesh and the corridor is found like in Algorithm 13, instead of $\boldsymbol{V}_{aux}$ vertices of incident triangles are used.

In both algorithms, the desired clearance (width) of the corridor is only approximation due to discretization of the triangular mesh. It is not an issue, because the corridor is used to create random samples around the auxiliary path.

The temporal goal q_t of the RRT-Path algorithm is substituted by the temporal point p_t in the RRT-Path^{ext} algorithm. The distance map to the goal is used to maintain p_t along the auxiliary path. The map is a set of the shortest distances from each vertex of the mesh to the goal vertex. If a new configuration q_{new} is added to the tree, the point p_t is updated by Algorithm 15. The idea of the distance to the goal is similar to the cost of the node introduced in [256].

A new random temporal goal q_t is randomly sampled inside the corridor around p_t.

Algorithm 15: Maintenance of temporal point p_t along the auxiliary path

Input: $\mathcal{T} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{T})$ - a triangular mesh of $\mathcal{W}$
Input: $distance\_map$ - all shortest path from each vertex $v \in \boldsymbol{V}$ to the goal
Input: $\boldsymbol{P} = (\boldsymbol{V}_{aux}, \boldsymbol{E}_{aux}, \boldsymbol{T}_{aux})$ - auxiliary path
Input: q_{new} - a new configuration added to the tree
Output: p_t - new temporal point at the auxiliary path

$p_{new} \leftarrow$ *two dimensional* (x, y) *projection of* q_{new}
$T \leftarrow$ *locate a triangle for* p_{new} *such that* $p_{new} \in T$
$\boldsymbol{d} \leftarrow \emptyset$`// set of distances from` p_{new} `to goal over vertices of` T
foreach $v \in T$ **do**
  $\boldsymbol{d} \leftarrow \boldsymbol{d} \cup \{|(p_{new}, v)| + distance\_map(v)\}$
$d \leftarrow \min \boldsymbol{d}$ `// find minimal distance to the goal`
$p_t \leftarrow$ *find new point at the path* $\boldsymbol{P}$ *that is in* d *distance from the goal*

A neighbourhood of p_t along the path is defined by two parameters called horizons: backward (h_b) and forward (h_f). The sampling procedure is depicted in Algorithm 16.

Algorithm 16: Random sampling of temporal goal g_t in an auxiliary corridor

Input: $\boldsymbol{P} = (\boldsymbol{V}_{aux}, \boldsymbol{E}_{aux}, \boldsymbol{T}_{aux})$ - an auxiliary path
Input: p_t - temporal point at the auxiliary path
Input: h_b, h_f - forward and backward horizon
Output: q_t - random temporal goal

$p_b \leftarrow p_t \ominus h_b$ `// aux. path point in` h_b `distance from` p_t `(or first pt.)`
$p_f \leftarrow p_t \oplus h_f$ `// aux. path point in` h_f `distance from` p_t `(or last pt.)`
$p_r \leftarrow$ *select random point at the auxiliary path between* p_b *and* p_f
$v \leftarrow$ *the closest vertex to* p_r, $v \in \boldsymbol{V}_{aux}$
$\boldsymbol{T} \leftarrow \{T | T \in \boldsymbol{T}_{aux}$ *and* T *is associated to* $v\}$
$T \leftarrow$ *select random triangle* T *according to area of triangles*
$p \leftarrow$ *random point inside the triangle* T
$q_t \leftarrow p$ `// use only (x,y)`

Even though the proposed sampling procedure improves generation of a random configuration, the configuration can be still undesirably close to obstacle, from which any other configuration is in collision. If such a configuration is selected repeatedly (as q_{near}), the growing process is stucked like in the RRT–Path. Inspired by viability concept [141], state-space search technique is utilized in the RRT–Path^{ext}. During the tree expansion all possible feasible configurations are generated for q_{near}. These configurations are preserved and if the same q_{near} is selected again, only not already used configurations are considered for the expansion. In the case q_{near} does not have any available configurations, the parent configuration is considered for the expansion in similar manner. This technique represents backtracking mechanism in a state-space search algorithm. Also instead of just one q_{near} set of the nearest configurations can be used like in [256].

Finally, to find multi-goal trajectory, which can be possible closed (e.g. a watchman route), it is necessary to split the trajectory into several parts. Such trajectories have to be connected in order to get a final smooth path. Connection of two trajectories is also necessary in the Bidirect variant of the RRT, in which two trees are grown, one from the

start and another one from the goal. The problem is easier for the multi-goal path, because the path is continuous and trajectories can be found sequentially. The goal configuration of a trajectory can be used as the initial configuration of the next part of the final trajectory. Because a final configuration can be in colliding direction with an obstacle, the trajectory is rather found for a goal that is more forward at the route.

More formally a path is a sequence of points $(p_0, p_1, \ldots, p_n)$ in $\mathcal{C}_{free}$. The path is split to several consecutive parts $(\boldsymbol{p}_{0,i,j}, \boldsymbol{p}_{i,j,k}, \ldots, \boldsymbol{p}_{l,m,o}, \boldsymbol{p}_{m,n,n})$, where $\boldsymbol{p}_{i,j,k} = (p_i, \ldots p_j, \ldots, p_k)$ and n is the number of path points. Each part overlaps with next part, except the final part. For a closed path $p_0 = p_n$, but the final configuration is not same as the start configuration, because the RRT algorithm is stopped if the trajectory is in a sufficient distance to the goal. Here, it is assumed that the robot stops at the final goal.

The RRT–Path^{ext} algorithm is based on the RRT–Path algorithm, and the schema of both algorithms is same. The extensions and differences are summarized in the following list:

- A triangular mesh of the robot workspace $\mathcal{W}$ is used as the supporting structure.
- The auxiliary path consists of the sequence of mesh vertices and corridor, which is a set of mesh triangles.
- The temporal point p_t at the auxiliary path is updated according to estimation of the tree distance to the goal. The triangular mesh is used to find the distance map to the goal. The distance map supports fast estimation of the distance from new configuration to the goal.
- The corridor is used to sample new random configuration around p_t, a neighbourhood of p_t is considered.

The multi-goal motion planning based on trajectory partitioning can be used with the RRT–Path algorithm, but the main advantage of the RRT–Path^{ext} is fast convergence for paths, which are incident with obstacles, e.g. a solution of the TSP on the complete visibility graph.

8.2.3 Experimental Results

The performance of the proposed RRT–Path^{ext} has been experimentally evaluated for a set of random trajectories in the map *jh*, which contains many narrow passages, entrances to rooms. The real computational requirements of the algorithm are crucial for its application in the adaptation procedure, therefore required computational time is compared with the approximation of the shortest path described in Section 5.2.4. Two variants of the approximation over vertices of convex cells are examined: with and without consideration of the incident vertices of edges crossed by the direct line segment from the start to the goal point, the algorithms are referred as *SP* and *SP-obstacles*. The second variant is used to find an initial auxiliary path, which is used in Algorithm 14. The examined set of motion problems has been created as 1000 pairs of random initial and goal positions of the robot inside free space of the map *jh*.

Because of intention to used the proposed RRT–Path^{ext} algorithm in the combination of SOM as a trajectory generator, which is expected to be computationally intensive, a point robot is assumed. It allows to used the point-location algorithm to test if a random configuration is in $\mathcal{C}_{free}$, which reduces the required computational time of the RRT–Path^{ext} two times in comparison to the library RAPID [110]. The nearest configuration of the tree to the new random configuration is found by the MPNN library [279].

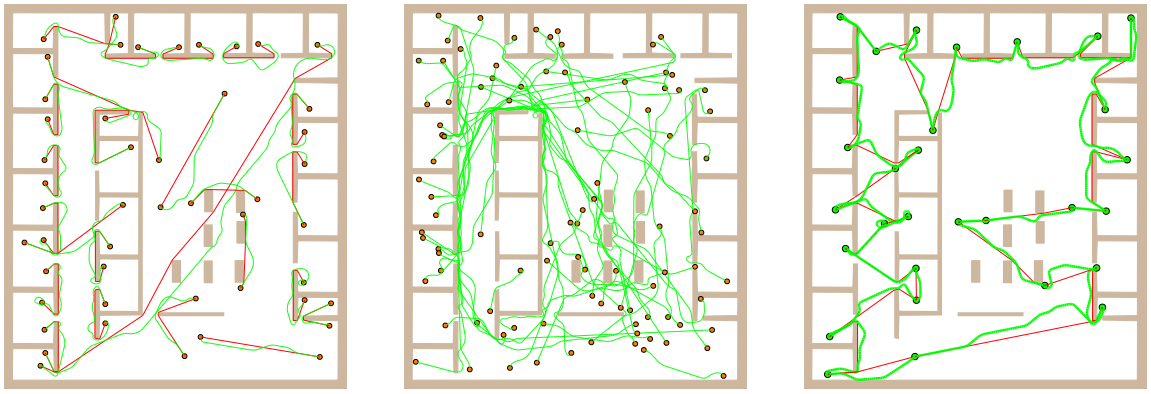
The robot with differential nonholonomic drive with maximal forward velocity 0.6 m.s^{-1} and with wheel radius 0.05 m is assumed. Wheels are at distance 0.3 m, that means free space of the map represents shrunk free space about more than fifteen centimeters. The robot is controlled by the radial wheel velocities and only in the forward direction, the set $\mathcal{U}$ has 24 different control inputs. The tree edge corresponds to the time interval of one second and it is discretized into five samples. The supporting triangular mesh to determine the corridor along the auxiliary path consists of 2506 vertices and 4536 triangles. Associated triangles to the auxiliary path are at distance 1 m, the forward horizon is 1 m and backward horizon is 0.2 m. A found trajectory is sampled per 0.1 m to form a polygonal path. The minimal required distance of the planned trajectory to the desired goal δ has been set to 0.15 m.

Experimental results are presented in Table 8.1. The RRT–Path^{ext} algorithm has been executed for several maximal number of steps K. The presented values represent average values for 1000 paths. The column *Time* denotes the average required computational time to find one path, the column *Length* is the average length of the found path. The fourth column denotes speed of the algorithm in the number of found paths/trajectories per one second. Next two columns are the average number of tree nodes and the average number of steps per found path. The last column shows the number of iterations, for which a path is not found in the maximal number of steps K. All algorithms have been implemented in C++, compiled by the G++ 4.2 with -O2 optimization flag and executed within the same computational environment using a single core of the Athlon X2 at 2 GHz CPU, 1 GB RAM running FreeBSD 7.1. Examples of found trajectories along auxiliary paths are depicted in Figure 8.4.

Algorithm	**Time** [ms]	**Length** [m]	**Found paths per second**	**Tree size**	**No. Steps**	**No. Fails**
SP	8 μs	13.3	125 000	-	-	0
SP-obstacles	13 μs	13.2	77 000	-	-	0
RRT-Path^{ext}, $K = 400$	30	13.8	33	63	69	26
RRT-Path^{ext}, $K = 1000$	31	14.0	33	69	79	13
RRT-Path^{ext}, $K = 2000$	31	14.1	32	71	87	4
RRT-Path^{ext}, $K = 5000$	31	14.1	32	70	85	0
RRT-Path^{ext}, $K = 10000$	31	14.1	32	67	91	0

Table 8.1: Performance of the RRT–Path^{ext} algorithm.

The proposed RRT–Path^{ext} algorithm provides thirty trajectories per seconds, which seems be sufficient for real-time planning, but it is still significantly slower than the approximation of the shortest path. According to real computational time of the algorithm it can be expected that possible solution of the multi-goal motion problem with the SOM procedure will be found in tens of minutes instead of units of seconds. Expected performance is not suitable for real application, but it allows preliminary experiments with combination of SOM and RRT approaches.



(a) trajectories for narrow passage problems

(b) trajectories for 50 random problems

(c) route is found in 420 ms and trajectory in 745 ms

Figure 8.4: Example of found trajectories by the RRT–Path^{ext} algorithm, the auxiliary path is in red, the found trajectory is in green: (**c**) found trajectories for solution of the TSP.

8.3 Adaptation procedure with trajectory consideration

This section describes a naïve approach to consider a trajectory generation during route optimization process and it should be considered as a feasibility study. The selected problem is the WRP and the MWRP for a group of cooperating robots, mainly because fix-points, which must be necessarily visited, are not prescribed, and solution (coverage) is affected by the found trajectory. In Chapter 7, the SOM algorithms for the WRP/MWRP have been proposed (Algorithm 10 and Algorithm 11) that are used in this feasibility study. The main idea of the trajectory consideration can be summarized in the following steps:

1. nodes are adapted to attraction points along the approximation of the shortest path,
2. the ring of nodes is used as the auxiliary path in the RRT–Path^{ext} algorithm to find a trajectory,
3. the trajectory is used to created a new ring of nodes by *ring regeneration* procedure,
4. the adaptation process is repeated for new presentation of triangles.

The ring regeneration means that nodes (weights) are placed at new positions without adaptation process[2], only 2D part of the configuration in $\mathcal{C}$ is used.

The adaptation procedure for the WRP with the RRT–Path^{ext} ring regeneration is depicted in Algorithm 17. The first part of the algorithm is identical to Algorithm 10. The RRT–Path^{ext} algorithm is utilized in the ring regeneration that is performed after i_{reg} steps, because in early steps of the adaptation rings are very small, see Figure 7.7. The ring of nodes is used to find a path by the approximation of the shortest path. The path is split to several overlapping parts and for each part a trajectory is found, which is then use to regenerate the ring. The ring regeneration requires a variable number of neurons in each adaptation step.

For new nodes created by the sampling of the trajectory a convex cell of the supporting convex partition have to be found in order to use the approximation of the shortest path.

[2]This technique has been used in the algorithm to solve the TSP with hierarchy of cities [88], in which a ring of nodes is firstly adapted to small set of cities, then the number of nodes is increased and new adaptation is performed for a problem with higher number of cities. The procedure reduces the required computational time while the quality of solution is worse only about units of percents.

Algorithm 17: WRP - SOM algorithm with the RRT–Path^{ext} ring regeneration

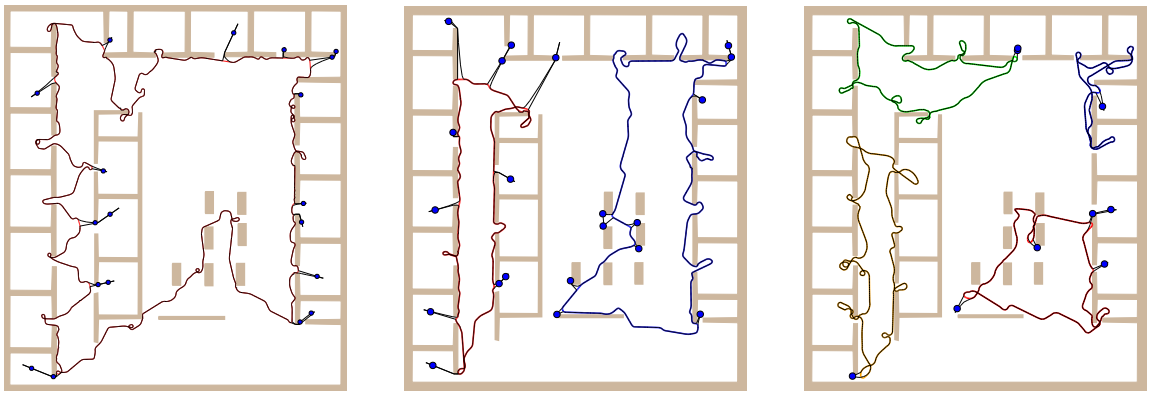
Input: $\mathcal{W}$ - map, polygonal representation of the robot workspace
Input: $\mathcal{T} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{T})$ - a triangular mesh of $\mathcal{W}$
Input: $\boldsymbol{P}$ - set of convex polygons associated to triangles (convex cover of $\mathcal{W}$)
Input: (d, G, μ, α) - parameters of SOM
Input: i_{reg} - step from when the ring regeneration is performed

```
r ← initialization                          // initial creation of ring of nodes
i ← 0                                       // set the adaptation step counter
repeat
    I ← ∅                                   // set of inhibited nodes
    T_c ← triangles covered by the current ring r
    foreach T ∈ Π(T) do                     // select T from a random permutation of T
        if T ∉ T_c then
            p_a ← centroid(T)               // attraction point
            ν* ← select winner node to p_a, ν* ∉ I   // approx. the shortest path
            P_c ← {all associated convex polygons to T}
            if ν* ∉ P, P ∈ P_c then
                adapt(ν*, p_a)
            T_c ← T_c ∪ {T|T ∈ P, P ∈ P_c}
            I ← I ∪ {ν*}                    // inhibit winner node
    if i ≥ i_reg then
        path ← ∅
        foreach {ν_i, ν_{i+1}} ∈ r do
            path ← (path, path(ν_i, ν_{i+1}))   // where path constructs path in W
        (p_0, p_1, ..., p_n) ← split path into several overlapping consecutive parts
        q_s ← ν_0                           // initial configuration
        r ← ∅                               // a new ring
        foreach p_i ∈ (p_0, p_1, ..., p_n) do
            t ← rrt_path_ext(q_s, p_i)      // plan trajectory from q_s along p_i
            q_e ← select configuration from t before trajectory end
            r ← (r, sample(t(q_s, q_e)))    // add trajectory points as new nodes
            q_s ← q_e                       // used q_e as new trajectory initial configuration
    G ← (1 − α) · G                         // decrease the gain
    i ← i + 1
until (all triangles are covered)
```

The cell is found by the algorithm based on the interval trees [217]. It provides more than four thousands queries per millisecond in the used computational environment, which is sufficient in comparison to required computational time to get the trajectory.

The MWRP variant is pretty much same to the WRP algorithm, it differs only in maintenance of several rings, as it has been already shown in the previous SOM based algorithms.

Examples of found trajectories during the WRP and MWRP adaptation procedures are shown in Figure 8.5. Blue disks represent new winner nodes, the colored paths represent a regenerated ring from the previous adaptation step, the thin black paths are current rings.



(a) (b) (c)

Figure 8.5: Approximate solution of the WRP with the RRT-Path^{ext} ring regeneration.

8.4 Discussion

Three methods to create velocity profiles for a group of cooperating robots in multi-goal path planning problem have been presented:

- trajectory generation along a smooth path (spline curves or navigation functions from APF),
- trapezoidal velocity profile for the shortest-path roadmaps,
- velocity profile found by the RRT–Path^{ext} algorithm.

The proposed RRT–Path^{ext} performs well in environments with narrow passages, and it provides good real-time performance. However the used auxiliary path in the RRT–Path^{ext} is only planar, the experimental results show significant reduction in the required size of the tree. Probably the guided sampling can also be helpful for high dimensional $\mathcal{C}$.

The found trajectories of the solved WRPs are completely smooth, but they contain several unnecessary loops. The trajectories are not satisfactory, but important lessons have been learned. At first, the SOM adaptation procedure can be used for evolving trajectories. The SOM adaptation performs well in separation of trajectories into particular non-overlapping paths. Sufficient number of nodes have to be used to respect smoothness of the found trajectory with sufficient level of details. It also is necessary to consider only local improvements of found trajectory by the RRT algorithm.

Consideration of trajectory generation in the multi-goal path planning opens possible future planning improvements. Trajectories allow optimization of required mission time or needed energy, also the coordination can be explicitly addressed if positions of robots in time are known. In addition, uncertainty of robot localization can be considered during SOM adaptation process. According to the planned trajectory a localization error can be estimated and desired destination can be changed, e.g. a new alternate point can be generated in the proposed WRP algorithm to decreased expected pose error.

Chapter 9

Conclusion

9.1 Conclusion

The multi-goal path planning problem for a group of cooperating mobile robots in the inspection task has been studied in this thesis. The inspection task is considered in two main approaches formulated as well known problems. The decoupled approach consists of the AGP and the MTSP formulations, it is motivated by discrete sensing capabilities of mobile robots and high sensing cost. The second approach assumes continuous sensing and it is formulated as the MWRP. The proposed application of the SOM adaptation procedure provides a flexible tool to address both approaches. It finds approximate solutions of the problems within reasonable time. Moreover SOM has been applied to the generalized multi-goal path planning problem where a goal is represented by a set of points instead of a single point. The problem of the cooperation is formulated as optimization of the MinMax criterion. The coordination is not explicitly addressed during the optimization, but the SOM procedure provides non-crossing paths with high frequency. The motion planning problem considered as the trajectory generation problem is addressed by two methods: trajectory generation along the found paths and trajectory generation during optimization.

In addition to the multi-goal path planning algorithm, two new algorithms have been proposed for the related AGP (sensor placement problem) and motion planning problem, particularly the Boundary Placement (BP) algorithm and the RRT–Path^{ext} algorithm. Two main aspects can be found in these algorithms. Both algorithms are randomized, which allows consideration of various constraints, and in both algorithms the randomization process is guided by a priori knowledge represented as a geometrical structure. The BP algorithm uses the boundary and the RRT–Path^{ext} algorithm uses the auxiliary path with a corridor.

An efficiency of the proposed algorithms is based on the supporting geometrical structures. Supporting structures are essential to combine visibility problems, route optimization and motion planning techniques all together in an algorithm with reasonable computational requirements allowing application of the algorithms in mobile robotics. The combination of structures from the computational geometry enables application of the SOM procedure in a polygonal domain where length of the Geodesic path has to be used instead of the Euclidean distance. Difficulty of the SOM application to such problems has been found in literature, therefore the main contribution of the thesis can be considered in

the proposed combination of supporting structures and the SOM adaptation procedure. Besides, particular contributions are presented in the following summary of contribution.

- Finding Sensing Locations - the AGP or sensor placement algorithms
  - an application of the polygon filter technique to the CPP algorithm,
  - a set of suitable parameters of the modified CPP and the RDS algorithms based on the experimental results in the maps of real environments and a set of visibility ranges,
  - a new algorithm called the Boundary Placement, which outperforms the CPP and the RDS in the number of found sensing locations as well as in the required length of the tour visiting the found set of locations,
  - a post-optimization procedure to reduce the number of found sensing locations for a restricted visibility range.
- Multi-Goal Path Planning for Cooperating Mobile Robots - MTSP-MinMax
  - an application of the SOM adaptation procedure in a polygonal domain based on the approximation of the shortest path among obstacles,
  - a determination of the length of the ring as the length of the tour represented by the ring, which seems to increase quality of found solutions,
  - an experimental verification of the MTSP algorithms (variants of GENIUS and SOM) in a set of problems in a polygonal domain where cities are found as sensing locations for the given visibility range,
  - an algorithm for the multi-depot MTSP-MinMax based on the SOM procedure in a polygonal domain,
  - an experimental verification of the SOM procedure on a triangular mesh for a set of mesh densities,
  - an application of navigation functions in the SOM adaptation procedure,
  - an adaptation rule for a representative of AoI,
  - an algorithm for the generalized multi-goal path planning problem.
- Cooperative Inspection Task in a Polygonal Domain
  - an algorithm for the decoupled approach based on the solution of the AGP with the visibility range constraint and the MTSP-MinMax,
  - an algorithm for the cooperative inspection of segment sensing locations,
  - an algorithm for the cooperative inspection of convex areas of interest,
  - an algorithm for MWRP-MinMax with restricted visibility range,
  - a feasibility study of the MWRP-MinMax with restricted visibility range and kinodynamic constraints.
- Motion Planning - Trajectory Generation
  - experimental results of trajectory consideration in the route optimization, an influence to the solution quality,
  - a comparison of performance of the RRT–Path algorithm with various auxiliary paths,
  - a new motion planner called the RRT–Path^{ext}, which is able to solve the multi-goal motion planning problem.

Regarding the presented contributions the goals of the thesis are fulfilled.

9.2 Future Work

The proposed approaches addressing variants of the multi-goal path planning and cooperative inspection consist of several components, which are particular algorithms or supporting structures. These components can be reconsidered to improve overall performance of the algorithms and detail discussion of particular possible improvements have been presented in each relevant chapters, therefore this section is focused on overall directions of the future work. At first, three motivation can be identified for possible improvements:

1. quality of solution,
2. required computational time,
3. parameters settings.

A better solution does not necessary mean increased computational time, which can be an important aspect of eventual improvements of supporting structures. A more suitable geometrical structure can improve quality of solutions and reduced the computational time, e.g. consideration of the boundary in the BP. Also less parameters are more practical and it is great advantage if an algorithm is self-tunable.

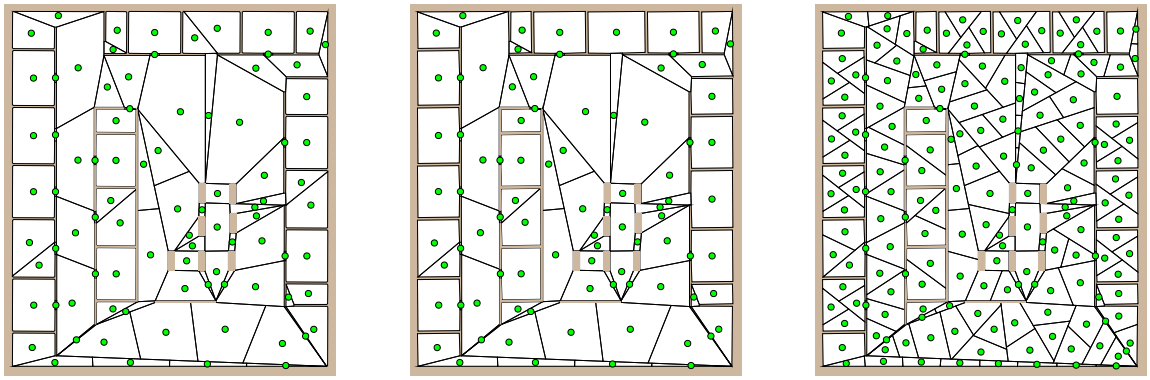
The presented results of the proposed SOM adaptation procedures in a polygonal domain for variants of the inspection task and multi-goal planning demonstrate feasibility of the proposed approach and can be motivation for additional applications to another problems from the class of hybrid visibility problems. From the another point of view the proposed algorithms are inspiring to consider more complex constraints all together that is especially important in robotic applications. Possible future steps beyond presented approach can be in following directions:

- an explicit consideration of coordination, application of SOM in high dimensional input space to directly use nodes in $\mathcal{C}$,
- an extension to a 3D model of environment,
- a consideration of complex kinematic and kinodynamic constraints,
- a consideration of path planning with uncertainty,
- a solution for other cooperative visibility based task, e.g. the *Covert Navigation*,
- an application of the proposed WRP algorithm to the AGP or the *Vision Points Problem*,
- an application of ideas of the generalized multi-goal path planing approach to the *Touring a Sequence of Polygons Problem*,
- a consideration of the sensing and motion costs in the *View Planning Problem*.

Appendix A

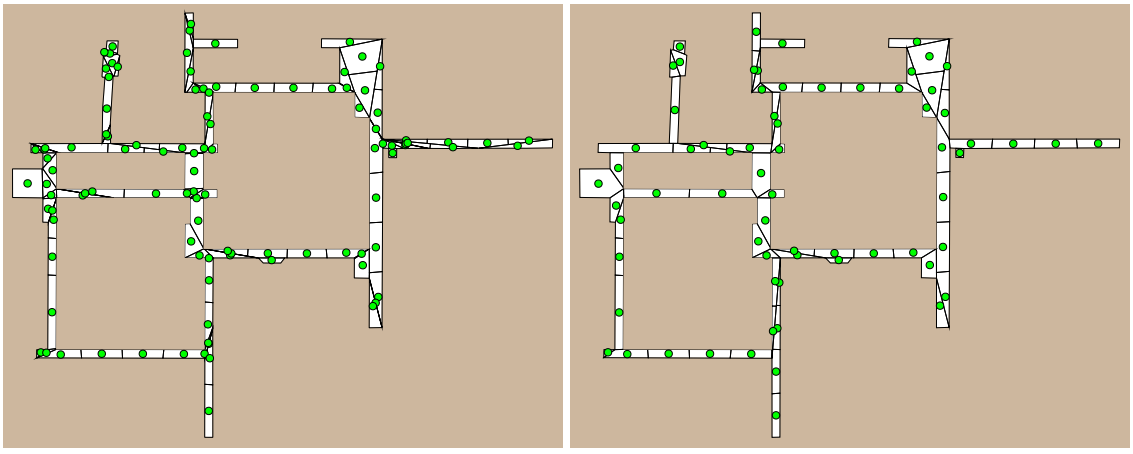
Algorithm Experiments – AGP

A.1 The CPP algorithm



(a) d=5 m, k=0 cm (b) d=5 m, k=5 cm (c) d=2 m, k=5 cm

Figure A.1: Found guards by the CPP algorithm, map *jh* for the visibility range d and the relevance k.



(a) k=0 cm (b) k=5 cm

Figure A.2: Found guards by the CPP algorithm, map *pb* and visibility range 10 m.

k [cm]	**Size** [m × m]	**Area** [m^2]	N_V
0	20.63 × 23.24	459.61	278
1	20.63 × 23.24	459.61	278
2	20.62 × 23.23	455.17	198
3	20.60 × 23.23	455.11	197
4	20.60 × 23.23	454.99	196
5	20.60 × 23.23	454.99	196

(a) map *jh*

k [cm]	**Size** [m × m]	**Area** [m^2]	N_V
0	39.72 × 46.84	730.05	138
1	39.72 × 46.84	731.24	98
2	39.72 × 46.84	730.44	78
3	39.72 × 46.84	730.80	76
4	39.72 × 46.84	730.38	75
5	39.72 × 46.84	730.67	74

(b) map *ta*

k [cm]	**Size** [m × m]	**Area** [m^2]	N_V
0	133.28 × 104.81	1456.25	176
1	133.28 × 104.81	1456.59	127
2	133.28 × 104.77	1454.84	97
3	133.28 × 104.77	1453.81	90
4	133.28 × 104.77	1453.81	90
5	133.28 × 104.77	1453.33	89

(c) map *pb*

Table A.1: Size of maps and the number of vertices N_V after applying the polygon filtering technique with particular relevance measure k.

d [m]	**Number Guards** k=0	k=1	k=5	**Tour Length** [m] k=0	k=1	k=5
inf	79	79	77	197	197	193
10.0	80	80	78	198	198	195
5.0	87	87	85	208	208	204
4.0	91	91	89	212	212	208
3.0	102	102	100	219	219	216
2.0	178	178	180	294	294	295
1.5	284	284	282	360	360	359
1.0	567	567	563	488	488	485

(a) map *jh*

Number Guards k=0	k=1	k=5	**Tour Length** $[m]$ k=0	k=1	k=5
57	38	30	232	211	198
58	39	31	232	213	197
83	67	58	273	264	255
101	84	72	296	284	275
145	128	116	330	320	315
263	239	221	422	413	401
434	413	395	527	523	518
916	905	865	757	755	743

(b) map *ta*

Number Guards k=0	k=1	k=5	**Tour Length** [m] k=0	k=1	k=5
81	45	41	612	543	534
107	73	69	636	619	610
172	127	122	707	688	679
199	153	149	731	717	713
279	228	223	785	774	771
483	399	394	901	881	876
859	771	762	1090	1108	1068
1852	1763	1765	1544	1521	1521

(c) map *pb*

Table A.2: The CPP algorithm performance for selected filtered maps (relevance measure k in centimeters) and particular visibility range d.

A.2 The RDS algorithm

Guards Ratio

m	**Maps**			**Visibility ranges** [m]						
	jh	**ta**	**pb**	**inf**	**10.0**	**5.0**	**4.0**	**3.0**	**2.0**	**1.5**	**1.0**
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
5	0.88	0.89	0.87	0.82	0.87	0.89	0.88	0.88	0.89	0.91	0.91
10	0.87	0.88	0.86	0.78	0.86	0.89	0.87	0.87	0.89	0.90	0.90
25	0.87	0.89	0.85	0.77	0.83	0.90	0.87	0.88	0.89	0.91	0.92
50	0.88	0.89	0.85	0.74	0.83	0.90	0.88	0.89	0.90	0.91	0.93
75	0.88	0.91	0.86	0.77	0.84	0.90	0.88	0.91	0.91	0.92	0.94
100	0.89	0.91	0.86	0.76	0.83	0.91	0.89	0.91	0.92	0.92	0.94

(a) Guards Ratios

Length Ratio

m	**Maps**			**Visibility ranges** [m]						
	jh	**ta**	**pb**	**inf**	**10.0**	**5.0**	**4.0**	**3.0**	**2.0**	**1.5**	**1.0**
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
5	0.93	0.96	0.95	0.92	0.95	0.95	0.95	0.94	0.95	0.96	0.95
10	0.91	0.95	0.95	0.89	0.94	0.95	0.93	0.93	0.95	0.95	0.95
25	0.90	0.95	0.95	0.87	0.91	0.94	0.93	0.93	0.95	0.96	0.95
50	0.89	0.95	0.95	0.86	0.91	0.93	0.92	0.95	0.96	0.95	0.96
75	0.89	0.96	0.95	0.86	0.90	0.93	0.93	0.95	0.96	0.96	0.96
100	0.89	0.95	0.95	0.85	0.90	0.93	0.92	0.95	0.96	0.95	0.96

(b) Length Ratios

Table A.3: Experimental results of the RDS algorithm, average ratios of the number of guards and length of the tour for the particular map and for the visibility range according to the number of random samples m. The ratios are computed according to average values for m=1. Standard deviations are very similar for all maps and the number of samples and they increase with shorter visibility range and decrease with increasing number of samples. The highest value of the ratio between number of guards and its standard deviation is for the unrestricted visibility range and the values are around 10%. The ratio decreases and it is around 1% for the lowest visibility range one meter. It is similar for the lengths of tour, the highest ratio is less than 7% for the unrestricted visibility range and it is less than 1.3% (typically 0.7%) for the visibility range one meter.

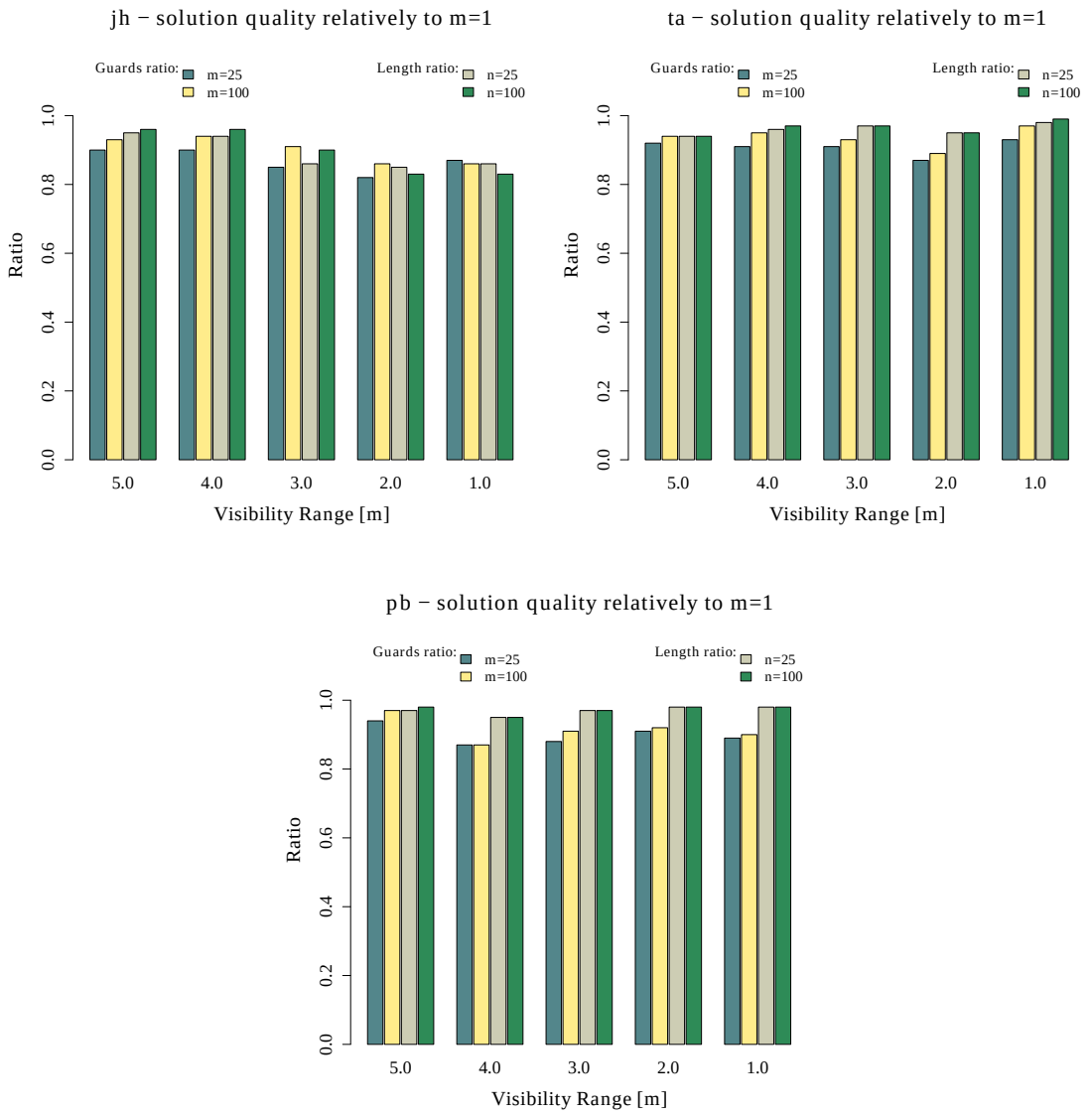
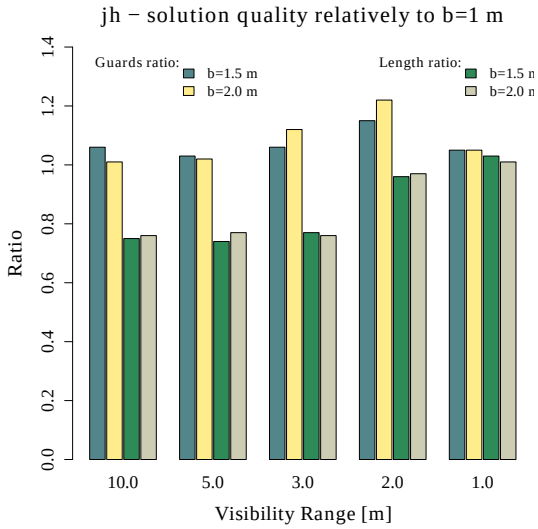
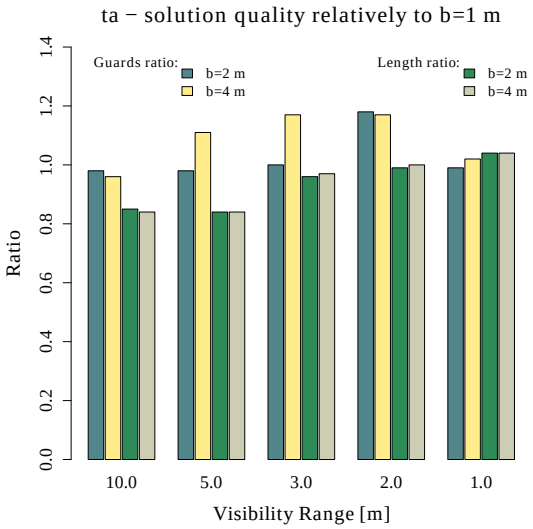


Figure A.3: Quality of solutions found by the RDS algorithm according to the number of samples m, the first two columns represent guards ratios for m=25, resp. m=100 and the next two columns represent the length ratios.

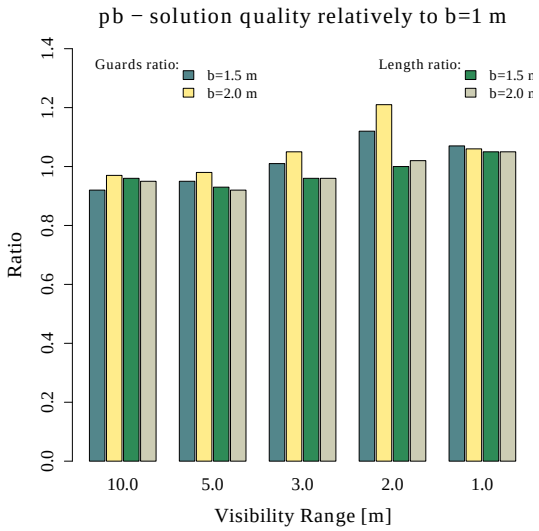
A.3 The BP algorithm



(a) map *jh*

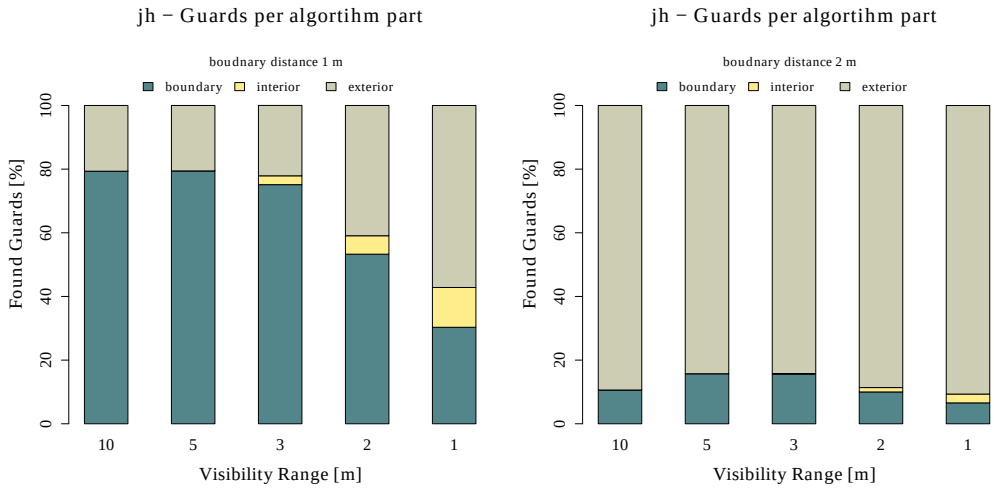


(b) map *ta*



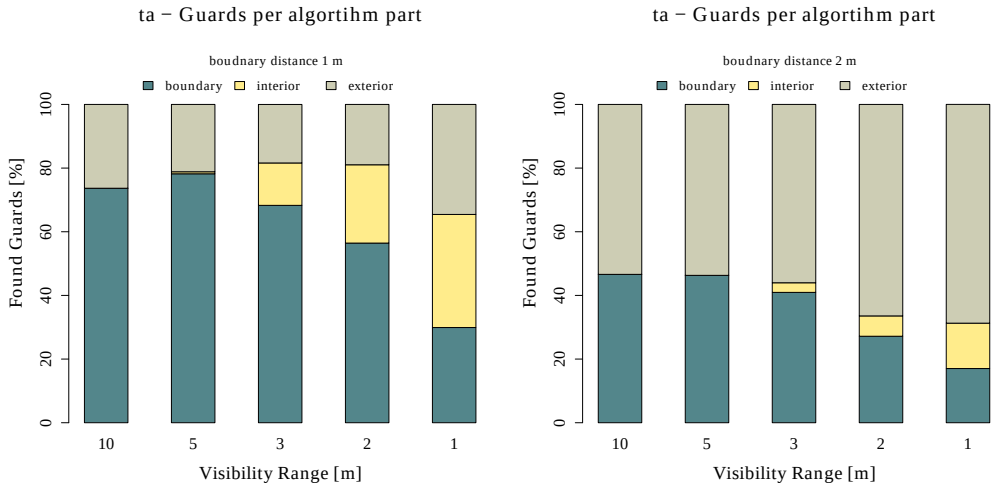
(c) map *pb*

Figure A.4: Performance of the BP algorithm according to the boundary value b.



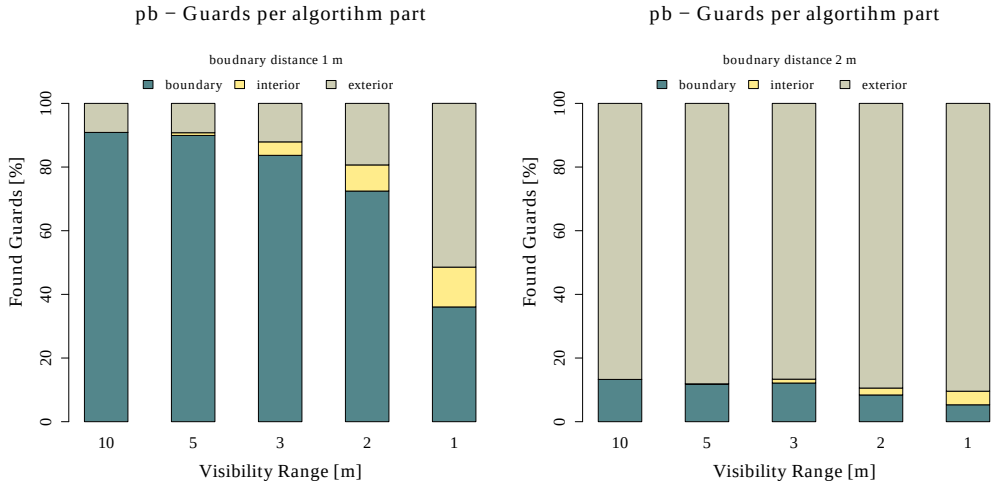
(a) b=1 m

(b) b=2 m



(c) b=1 m

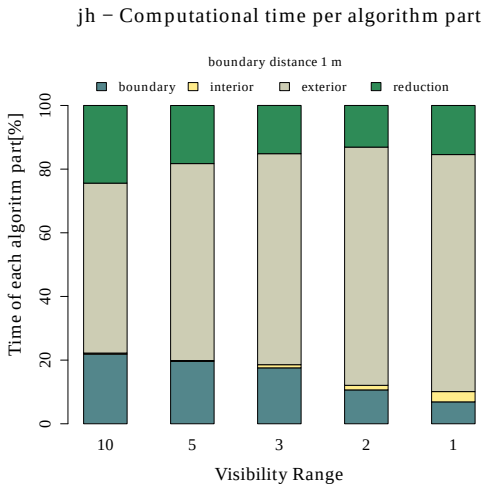
(d) b=2 m



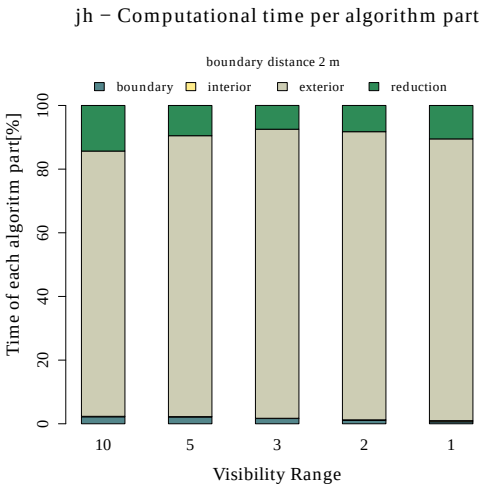
(e) b=1 m

(f) b=2 m

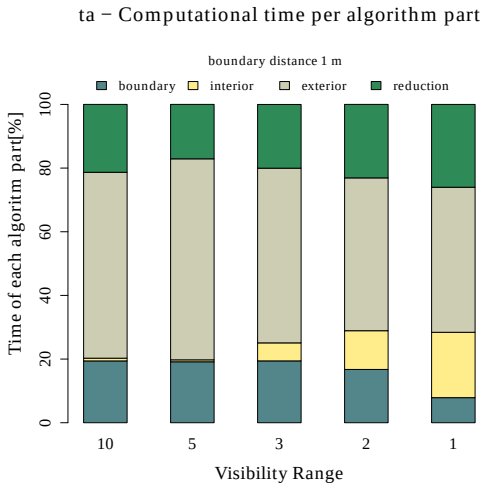
Figure A.5: Found guards in particular part of the BP algorithm.



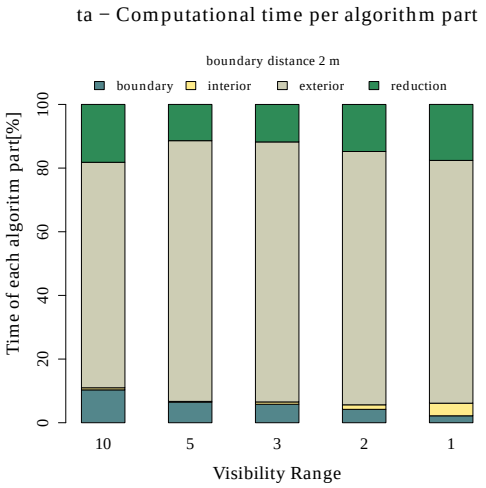
(a) b=1 m



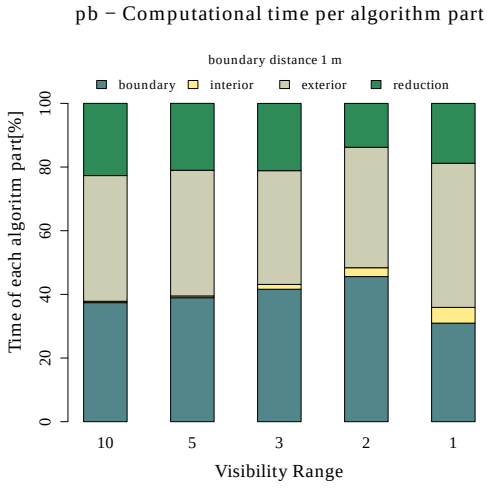
(b) b=2 m



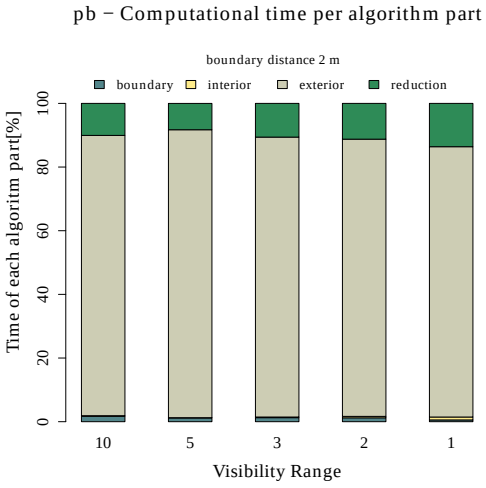
(c) b=1 m



(d) b=2 m



(e) b=1 m



(f) b=2 m

Figure A.6: Computational time spent if particular part of the BP algorithm.

A.4 Algorithms Comparison

d	**No. Guards**			**Length** [m]		
[m]	**CPP**	**RDS**	**BP**	**CPP**	**RDS**	**BP**
inf	94	33	31	208	154	149
10.0	95	37	32	207	158	118
5.0	101	44	37	216	160	124
4.0	106	47	41	220	164	125
3.0	115	63	53	226	190	138
2.0	175	126	94	282	269	217
1.5	282	200	160	350	321	258
1.0	552	388	340	471	414	359

(a) map *jh*

No. Guards			**Length** [m]		
CPP	**RDS**	**BP**	**CPP**	**RDS**	**BP**
34	14	13	204	167	167
35	18	15	203	176	159
58	41	34	254	239	197
72	54	52	272	260	224
118	83	83	315	292	266
231	170	139	408	366	330
405	280	231	522	436	386
865	598	526	746	595	528

(b) map *ta*

No. Guards			**Length** [m]		
CPP	**RDS**	**BP**	**CPP**	**RDS**	**BP**
45	16	20	533	389	407
73	41	44	613	567	533
131	90	84	683	667	619
160	120	109	720	697	650
235	178	160	775	741	696
434	349	271	902	851	787
821	616	439	1116	1001	858
1809	1289	1067	1565	1346	1151

(c) map *pb*

Table A.4: Performance of the CPP, RDS and BP algorithms.

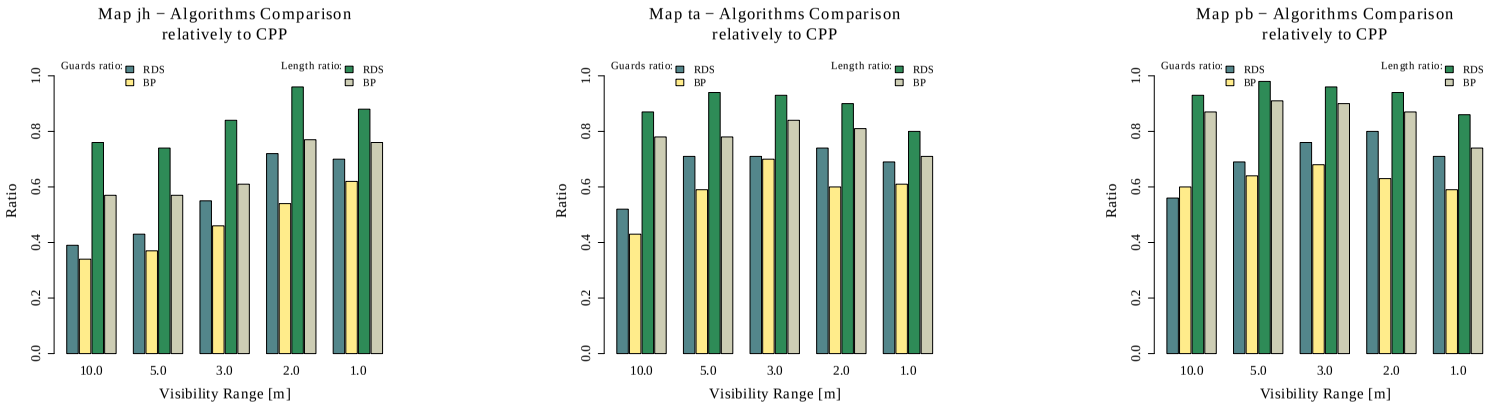
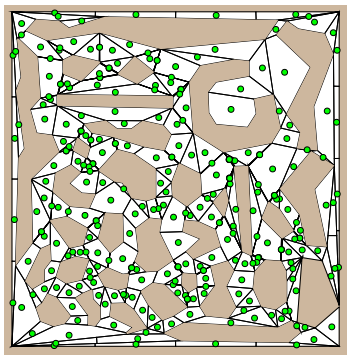
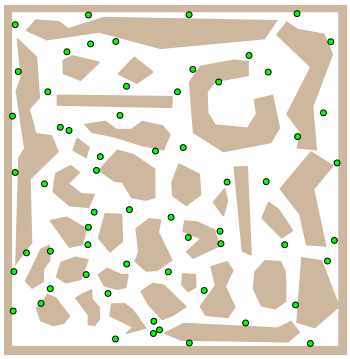


Figure A.7: Quality of found solutions, ratios relative to the CPP.

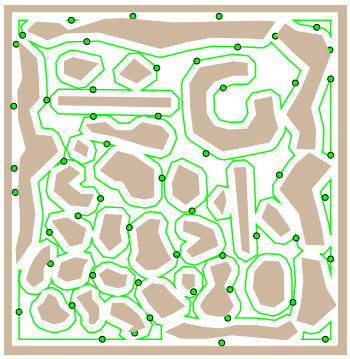
A.5 Other Environments



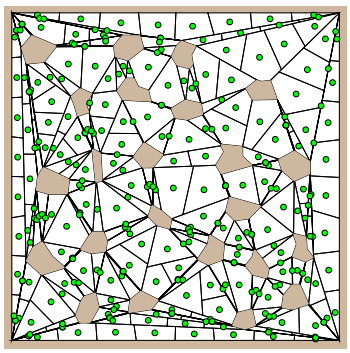
(a) map *dense*, d=4 m, CPP, G=285, L=270 m



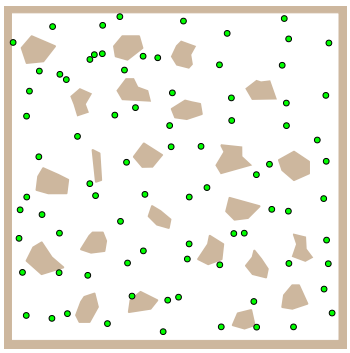
(b) map *dense*, d=4 m, RDS, m=25, G=61, L=181 m



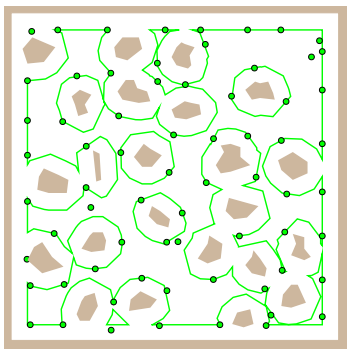
(c) map *dense*, d=4 m, BP, d=0.5 m, G=53, L=180 m



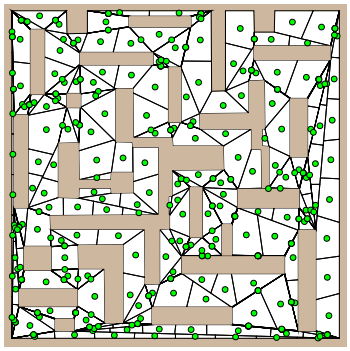
(d) map *potholes*, d=2 m, CPP, G=306, L=234 m



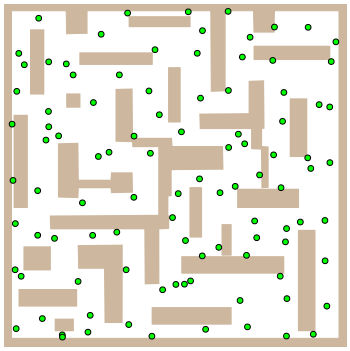
(e) map *potholes*, d=2 m, RDS, G=81, L=159 m



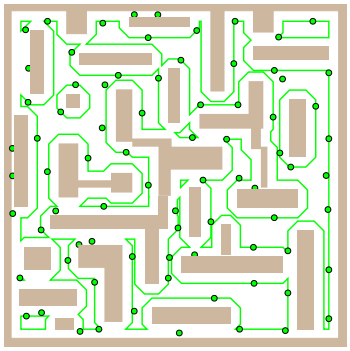
(f) map *potholes*, d=2 m, BP, d=1 m, G=68, L=155 m



(g) map *warehouse*, d=4 m, CPP, G=361, L=496 m



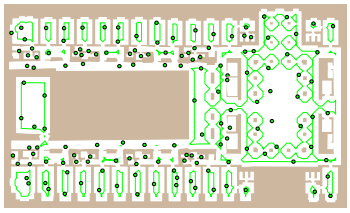
(h) map *warehouse*, d=4 m, RDS, m=25, G=99, L=383 m



(i) map *warehouse*, d=4 m, BP, d=1.2 m, G=78, L=363 m

(j) map *h2*, d=5 m, CPP, G=361, L=1353 m

(k) map *h2*, d=5 m, RDS, m=25, G=99, L=1132 m



(l) map *h2*, d=5 m, BP, b=1.5 m, G=78, L=887 m

Figure A.8: Solutions for the maps *dense*, *potholes*, *warehouse* with the square ground plan with size 21, 20 and 40 meters, and map *h2* of real building with size approximately seventy times forty meters, G is the number of guards and L is the length of the path.

Appendix B

Algorithm Experiments – TSP/MTSP

B.1 Testing Environments

Name	Size [m × m]	Area [m^2]	No. Vertices	No. Holes	No. Convex Polygons
jari	4.5 × 4.9	20	48	1	14
complex2	20.0 × 20.0	322	40	3	21
m1	4.8 × 4.8	20	51	4	26
m2	4.8 × 4.8	15	51	6	20
map	4.8 × 4.8	14	68	8	36
potholes	20.0 × 20.0	367	153	23	75
rooms	20.0 × 20.0	351	80	0	33
a	8.9 × 14.1	71	99	6	22
dense	21.0 × 21.5	299	288	32	150
m3	4.8 × 4.8	17	308	50	120
warehouse	40.0 × 40.0	1192	142	24	83
jh	20.6 × 23.2	455	196	9	77
pb	133.3 × 104.8	1453	89	3	41
ta	39.6 × 46.8	731	74	2	30
h2	84.9 × 49.7	2816	2062	34	476

Table B.1: Testing environments with obstacles.

Name	No. Cities
jari	6
complex2	8
m1	11
m2	12
map	18
potholes	18
rooms	21
a	22

(a) small set

Name*	No. Cities
$dense_4$	53
$potholes_2$	68
$m3_1$	71
$warehouse_4$	79
jh_2	80
pb_4	105
ta_2	141
$h2_5$	168

(b) middle set

Name*	No. Cities
$potholes_1$	282
jh_1	356
$pb_{1.5}$	415
$h2_2$	568
ta_1	574

(c) large set

Table B.2: Set of problems for environments with obstacles.

*Subscript denotes visibility range in meters.

Name	Coordinates		Depot Variant
dense	10.5,	12.0	depot
m3	0.25,	25.0	depot
potholes	24.9,	30.2	depot
pb	65.5,	44.0	depot
ta	30.5,	27.0	depot
warehouse	21.0,	19.5	depotA
warehouse	15.0,	10.0	depotB
jh	10.0,	0.5	depotA
jh	10.5,	11.0	depotB
h2	57.5,	-31.4	depotA
h2	80.4,	-46.6	depotB

Table B.3: Depot coordinates for particular maps, in map coordinate system converted to meters and rounded to decimals.

B.2 TSP Results

SP	MS **[%]**	$\#s$	LR	s_{LR}	TR
pa	0	64	1.02	0.035	1.76
va-0	14	79	1.04	0.045	1.24
va-1	0	63	1.02	0.039	1.58
va-2	0	64	1.02	0.046	1.75
va-3	0	63	1.02	0.034	1.77

(a) *small* problem set, *error* stop condition

SP	MS **[%]**	$\#s$	LR	s_{LR}	TR
pa	0	28	1.13	0.147	1.32
va-0	0	27	1.12	0.139	1.00
va-1	0	27	1.12	0.163	1.23
va-2	0	27	1.13	0.170	1.28
va-3	0	28	1.12	0.148	1.32

(b) *small* problem set, *unique* stop condition

SP	MS **[%]**	$\#s$	LR	s_{LR}	TR
pa	0	83	1.05	0.038	3.28
va-0	45	128	1.09	0.043	1.81
va-1	0	84	1.05	0.047	2.60
va-2	0	84	1.05	0.044	3.16
va-3	0	84	1.05	0.046	3.27

(c) *middle* problem set, *error* stop condition

SP	MS **[%]**	$\#s$	LR	s_{LR}	TR
pa	0	61	1.05	0.045	2.43
va-0	1	63	1.08	0.051	1.00
va-1	0	61	1.05	0.042	1.95
va-2	1	62	1.05	0.046	2.39
va-3	0	61	1.05	0.046	2.44

(d) *middle* problem set, *unique* stop condition

SP	MS **[%]**	$\#s$	LR	s_{LR}	TR
pa	0	99	1.04	0.019	2.51
va-0	17	115	1.14	0.083	0.93
va-1	0	100	1.04	0.018	1.92

(e) *large* problem set, *error* stop condition

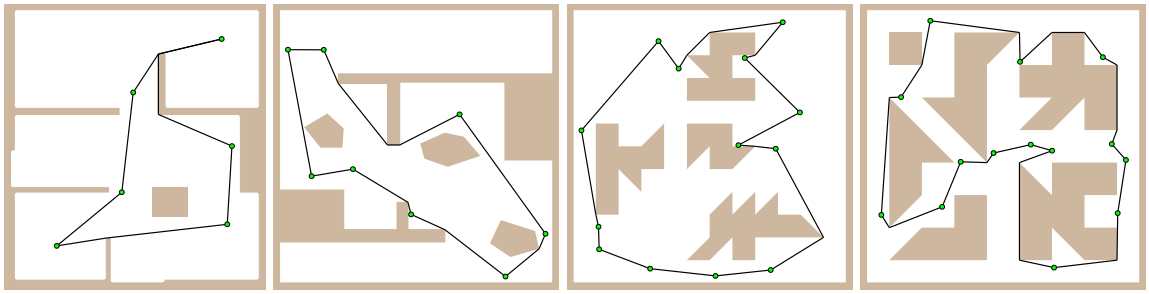
SP	MS **[%]**	$\#s$	LR	s_{LR}	TR
pa	3	91	1.04	0.020	2.24
va-0	38	125	1.14	0.080	1.00
va-1	4	91	1.04	0.019	1.73

(f) *large* problem set, *unique* stop condition

Table B.4: Approximation of the node–city path, the SP column is the path refinement variant.

Name	N_C	L_{opt} [m]	SOM *va-0*			SOM *va-1*			SOM *pa*		
			PDM	PDB	T [s]	PDM	PDB	T [s]	PDM	PDB	T [s]
jari	6	13.6	0.72	0.00	0.10	0.00	0.00	0.10	0.65	0.00	0.11
complex2	8	58.5	2.48	0.00	0.11	0.00	0.00	0.13	0.22	0.00	0.14
m1	13	17.1	4.50	0.00	0.15	0.44	0.00	0.18	0.18	0.00	0.21
m2	14	19.4	12.26	7.36	0.16	10.68	0.00	0.21	8.78	0.00	0.21
map	17	26.5	4.00	0.00	0.22	1.70	0.00	0.30	3.88	0.00	0.32
potholes	17	88.5	1.76	0.00	0.38	0.78	0.00	0.39	1.07	0.00	0.47
a	22	52.7	1.47	0.64	0.29	0.61	0.00	0.44	0.64	0.00	0.53
rooms	22	165.9	2.24	0.46	0.32	1.19	0.08	0.50	1.05	0.05	0.54
dense_4	53	179.1	14.09	6.97	2.75	15.34	7.68	2.92	11.98	7.00	3.62
potholes_2	68	154.5	12.84	8.97	3.48	5.10	3.03	3.94	5.45	3.79	5.38
m3_1	71	39.0	7.28	5.18	2.87	6.88	4.77	5.09	6.69	4.19	6.15
warehouse_4	79	369.2	11.17	5.22	4.79	6.30	2.28	5.29	6.89	3.32	6.63
jh_2	80	201.9	7.70	6.08	2.58	1.78	0.43	5.61	1.74	0.00	7.63
pb_4	104	654.6	4.75	2.18	3.62	0.56	0.14	7.78	1.41	0.03	8.50
ta_2	141	328.0	7.79	5.47	6.42	3.02	1.94	13.17	3.29	2.44	16.00
h2_5	168	943.0	2.77	1.83	37.23	1.73	1.18	39.24	1.68	1.02	53.97
potholes_1	282	277.3	30.17	27.08	31.16	6.23	4.73	72.15	6.08	4.65	105.38
jh_1	356	363.7	9.94	8.20	50.11	3.85	2.76	119.02	3.73	2.54	164.63
$\text{pb}_{1.5}$	415	839.6	8.05	4.82	59.61	2.00	1.05	133.72	2.38	1.28	152.11
h2_2	568	1316.2	10.67	7.47	330.21	2.57	1.90	428.65	2.24	1.67	603.96
ta_1	574	541.1	13.61	10.22	113.60	5.49	4.51	259.35	5.44	4.38	305.15

Table B.5: Detail TSP results, the number of cities is denoted as N_C, the SOM algorithm use the *error* stop condition.

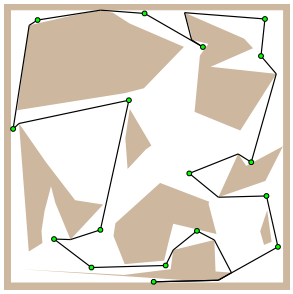


(a) map *jari*, N=6, L=13.6 m

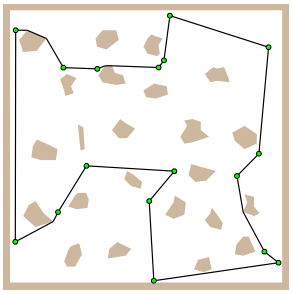
(b) map *complex2*, N=8, L=58.5 m

(c) map *m1*, N=13, L=17.1 m

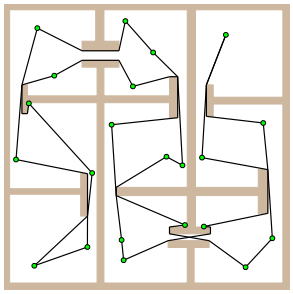
(d) map *m2*, N=14, L=19.4 m



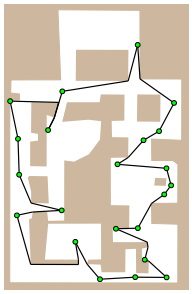
(e) map *map*, N=17, L=26.5 m



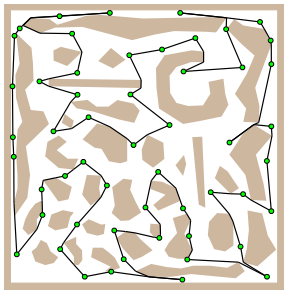
(f) map *potholes*, N=17, L=88.5 m



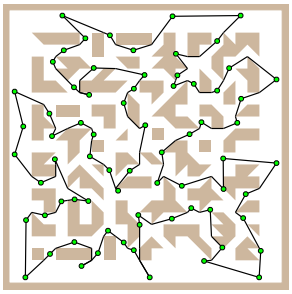
(g) map *rooms*, N=22, L=165.9 m



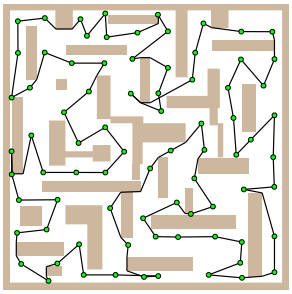
(h) map *a*, N=22, L=52.7 m



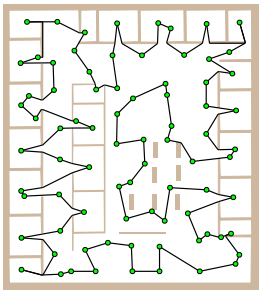
(i) map *dense*, N=53, L=191.7 m



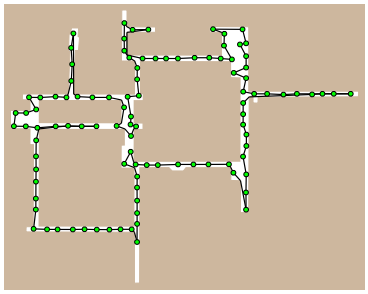
(j) map *m3*, N=71, L=40.7 m



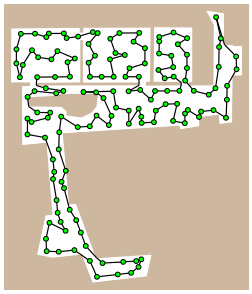
(k) map *warehouse*, N=79, L=381.4 m



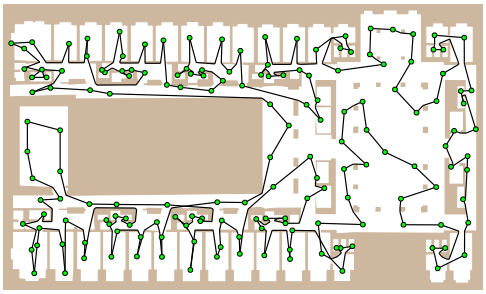
(l) map *jh*, N=80, L=201.9 m



(m) map *pb*, N=104, L=654.8 m



(n) map *ta*, N=141, L=336.0 m



(o) map *h2*, N=168, L=952.6 m

Figure B.1: Selected best solutions found by the SOM-*pa* variant with *error* stop condition, N is the number of cities and L is the length of the found tour.

B.3 MTSP Results

m	SOM-*nn*					SOM-*cc*				
	$\#s$	LR	CQ	CER	TR	$\#s$	LR	CQ	CER	TR
2	85	1.08	0.07	1.01	1.00	85	1.06	0.05	1.01	0.89
3	86	1.11	0.12	0.98	1.00	85	1.09	0.11	0.98	0.91
4	86	1.12	0.14	0.95	1.00	86	1.11	0.13	0.96	0.91
5	87	1.17	0.20	0.94	1.00	86	1.14	0.16	0.94	0.92

(a) SOM *error* stop condition

m	SOM-*nn*					SOM-*cc*				
	$\#s$	LR	CQ	CER	TR	$\#s$	LR	CQ	CER	TR
2	62	1.08	0.08	1.01	0.72	62	1.06	0.05	1.00	0.66
3	63	1.10	0.12	0.98	0.72	63	1.10	0.11	0.98	0.67
4	64	1.12	0.15	0.95	0.73	63	1.11	0.14	0.96	0.67
5	64	1.17	0.20	0.94	0.74	64	1.15	0.17	0.94	0.68

(b) SOM *unique* stop condition

m	*No.*	SOM-*nn*				SOM-*cc*			
	Iter	MS	s_{LR}	s_{CQ}	s_{CER}	MS	s_{LR}	s_{CQ}	s_{CER}
2	220	0.0	0.059	0.059	0.055	0.0	0.049	0.049	0.042
3	220	0.0	0.072	0.072	0.061	0.0	0.065	0.065	0.049
4	220	0.0	0.072	0.072	0.056	0.0	0.075	0.075	0.053
5	220	0.0	0.085	0.085	0.077	0.0	0.075	0.075	0.070

(c) SOM *error* stop condition, sample standard deviations

m	*No.*	SOM-*nn*				SOM-*cc*			
	Iter	MS	s_{LR}	s_{CQ}	s_{CER}	MS	s_{LR}	s_{CQ}	s_{CER}
2	220	0.0	0.060	0.060	0.058	0.0	0.051	0.051	0.041
3	220	0.0	0.074	0.074	0.063	0.0	0.070	0.070	0.053
4	220	0.5	0.075	0.075	0.063	0.5	0.068	0.068	0.054
5	220	0.0	0.080	0.080	0.080	0.5	0.081	0.081	0.073

(d) SOM *unique* stop condition, sample standard deviations

m	GENIUS - *quality*					GENIUS - *fast*				
	LR	CQ	CER	s_{LR}	s_{CER}	LR	CQ	CER	s_{LR}	s_{CER}
2	1.04	0.003	1.04	0.03	0.033	1.07	0.003	1.07	0.04	0.043
3	1.06	0.005	1.06	0.05	0.048	1.11	0.006	1.11	0.06	0.066
4	1.06	0.007	1.06	0.04	0.043	1.14	0.011	1.14	0.06	0.056
5	1.06	0.010	1.06	0.05	0.049	1.17	0.015	1.16	0.07	0.076

(e) GENIUS

Table B.6: MTSP results for the *middle* problem set.

m	SOM-*nn*					SOM-*cc*				
	$\#s$	LR	CQ	CER	TR	$\#s$	LR	CQ	CER	TR
2	100	1.05	0.05	1.00	1.00	100	1.03	0.04	0.99	0.94
3	100	1.07	0.09	0.96	1.00	100	1.04	0.07	0.97	0.95
4	100	1.09	0.12	0.94	1.00	100	1.06	0.11	0.94	0.96
5	100	1.09	0.16	0.91	1.00	100	1.05	0.12	0.91	0.96

(a) SOM *error* stop condition

m	SOM-*nn*					SOM-*cc*				
	$\#s$	LR	CQ	CER	TR	$\#s$	LR	CQ	CER	TR
2	91	1.05	0.06	0.99	0.90	93	1.03	0.04	0.99	0.88
3	102	1.07	0.10	0.97	1.02	90	1.04	0.07	0.97	0.85
4	104	1.09	0.12	0.94	1.04	101	1.06	0.10	0.94	0.96
5	105	1.09	0.16	0.91	1.05	100	1.06	0.14	0.91	0.95

(b) SOM *unique* stop condition

m	*No.*	SOM-*nn*				SOM-*cc*			
	Iter	MS	s_{LR}	s_{CQ}	s_{CER}	MS	s_{LR}	s_{CQ}	s_{CER}
2	140	0.0	0.038	0.038	0.039	0.0	0.031	0.031	0.025
3	140	0.0	0.076	0.076	0.054	0.0	0.045	0.045	0.032
4	140	0.0	0.073	0.073	0.050	0.0	0.062	0.062	0.042
5	140	0.0	0.090	0.090	0.065	0.0	0.064	0.064	0.054

(c) SOM *error* stop condition, sample standard deviations

m	*No.*	SOM-*nn*				SOM-*cc*			
	Iter	MS	s_{LR}	s_{CQ}	s_{CER}	MS	s_{LR}	s_{CQ}	s_{CER}
2	140	5.7	0.038	0.038	0.040	7.1	0.029	0.029	0.025
3	140	17.9	0.080	0.080	0.056	4.3	0.051	0.051	0.035
4	140	19.3	0.073	0.073	0.053	15.0	0.060	0.060	0.043
5	140	21.4	0.088	0.088	0.060	14.3	0.076	0.076	0.057

(d) *unique* stop condition, sample standard deviations

m	GENIUS - *quality*					GENIUS - *fast*				
	LR	CQ	CER	s_{LR}	s_{CER}	LR	CQ	CER	s_{LR}	s_{CER}
2	1.04	0.000	1.04	0.03	0.034	1.08	0.001	1.08	0.03	0.035
3	1.04	0.001	1.04	0.03	0.033	1.12	0.001	1.12	0.05	0.047
4	1.06	0.002	1.06	0.05	0.052	1.15	0.002	1.15	0.06	0.064
5	1.05	0.003	1.05	0.04	0.041	1.15	0.004	1.15	0.05	0.050

(e) GENIUS

Table B.7: MTSP results for the *large* problem set.

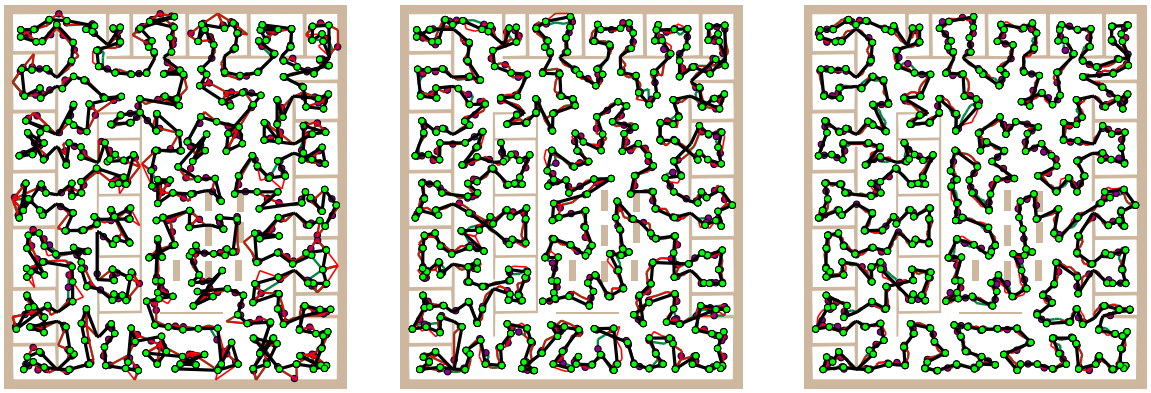
Problem	N_C	GENIUS-*quality*			GENIUS-*fast*			SOM-*nn*			SOM-*cc*		
		L_{ref}	PDM	T [s]	PDM	PDB	T [s]	PDM	PDB	T [s]	PDM	PDB	T [s]
dense$_4$	54	69.6	3.35	110.3	8.44	2.57	1.6	11.54	1.42	4.3	12.97	-0.41	3.8
potholes$_2$	69	58.3	3.03	185.1	7.44	3.38	1.9	7.23	0.76	6.1	11.58	3.22	5.6
m3$_1$	72	14.8	4.65	210.5	8.95	2.92	2.3	9.68	0.02	7.3	9.42	0.76	6.4
warehouse$_{4,A}$	80	141.4	3.15	232.1	7.20	0.65	2.7	8.01	-2.09	7.6	5.80	-0.88	6.9
warehouse$_{4,B}$	80	153.6	3.10	271.2	7.57	1.32	3.0	16.58	5.10	7.9	12.65	1.28	6.9
jh$_{2,A}$	81	80.5	5.88	282.9	11.22	6.12	3.2	18.65	15.86	8.4	14.37	1.54	7.9
jh$_{2,B}$	81	78.7	3.81	332.6	10.17	4.30	3.0	4.71	-1.01	8.7	5.18	-1.20	7.7
pb$_4$	105	281.4	4.45	575.0	10.61	2.58	5.5	10.99	6.50	9.8	10.15	2.57	8.9
ta$_2$	142	129.0	10.22	1 326.4	20.07	7.36	13.1	8.89	3.33	16.8	6.06	3.29	16.1
h2$_{5,A}$	169	350.9	8.28	1 635.5	14.52	4.96	18.2	4.43	-1.88	58.1	1.86	-1.80	52.8
h2$_{5,B}$	169	345.4	10.94	1 921.4	18.67	9.74	17.5	16.22	11.13	59.1	13.95	5.25	53.1
potholes$_1$	282	101.6	1.35	5 961.2	6.59	1.04	61.5	2.97	-2.47	108.5	3.07	-3.64	104.4
jh$_{1,A}$	357	133.0	3.89	8 397.0	12.77	9.10	86.8	12.78	7.01	184.3	8.93	3.62	172.4
jh$_{1,B}$	357	132.8	2.67	7 978.1	11.24	5.85	94.5	3.10	-1.91	171.9	2.28	-1.86	165.5
pb$_{1.5}$	416	338.7	5.96	15 727.8	13.45	3.49	224.5	4.98	2.31	163.3	5.95	2.27	154.7
h2$_{2,A}$	568	493.3	4.06	19 237.8	11.86	4.37	281.8	6.36	-3.75	627.4	1.30	-6.17	594.0
h2$_{2,B}$	568	501.6	5.06	24 803.6	11.42	5.20	411.6	-0.46	-3.79	625.6	-1.48	-4.65	604.1
ta$_1$	575	202.7	5.86	32 541.3	13.86	7.17	459.6	16.82	4.91	310.3	6.57	-0.13	296.2

Table B.8: Detail MTSP results for m=3, the number of cities is denoted as N_C, the SOM algorithm use the *error* stop condition.

B.4 SOM on a Triangular Mesh

#△	N_V	*jh*, 356 cities				*jh*, 36 cities			
		$\#s$	PDM	PDB	T [s]	$\#s$	PDM	PDB	T [s]
870	575	99	24.6	20.1	93.9	76	1.4	0.3	1.0
2 714	1 623	98	7.9	6.6	118.4	76	1.2	0.1	1.3
6 174	2 527	98	6.0	4.4	154.0	75	0.7	0.0	1.6
10 845	5 961	98	5.1	4.0	180.2	74	0.6	0.1	2.1
15 213	8 212	98	4.6	3.5	204.0	74	0.4	0.0	2.5
18 843	10 150	98	4.3	3.0	222.2	74	0.6	0.0	2.7
24 033	12 867	98	4.4	3.5	245.3	74	0.8	0.0	2.9
pa		97	3.7	2.5	164.6	75	0.6	0.1	1.8
va-0		98	9.9	8.2	50.1	75	1.1	0.1	0.7
va-1		97	3.9	2.8	119.0	75	0.9	0.0	1.3

Table B.9: Performance of the SOM adaptation procedure on the triangular mesh, $\#\triangle$ is the number of triangles, N_V is the number of graph nodes (cities are not included) and T is required computational time in seconds.



(a) 870 triangles, L=460 m (b) 2 714 triangles, L=394 m (c) 10 845 triangles, L=380 m

Figure B.2: SOM solutions of the TSP according to triangular mesh granularity, environment *jh* and 356 cities, L denotes the length of the found tour.

B.5 TSPLIB problems

Problem name	Depot index
berlin52	47
bier127	25
ch130	53
eil101	26
eil76	75
st70	5
rd100	46
rat195	97

(a) *tsplib-middle*

Problem name	Depot index
gil262	115
kroB200	57
lin318	157
rat575	287
rd400	289
u574	234

(b) *tsplib-large*

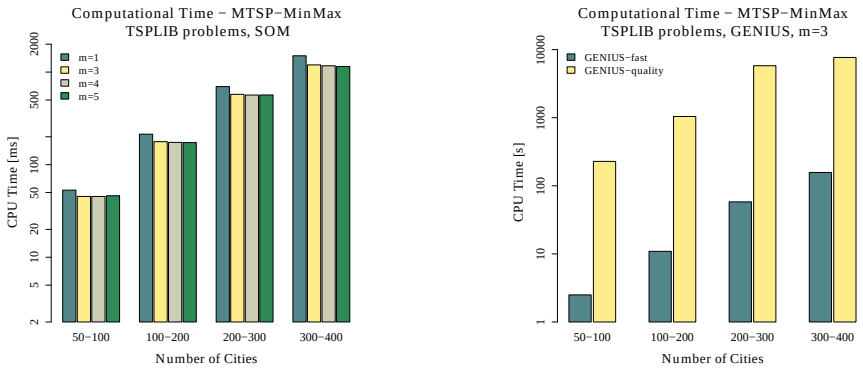
m		**SOM**				**GENIUS-*quality***			**GENIUS-*fast***			
	$\#s$	LR	CQ	CER	s_{LR}	LR	CQ	CER	LR	CQ	CER	s_{LR}
1	73	1.05	-	-	0.02	1.03	-	-	1.05	-	-	0.02
2	74	1.08	0.071	1.01	0.06	1.04	0.002	1.04	1.08	0.002	1.08	0.05
3	75	1.17	0.139	1.01	0.09	1.05	0.005	1.05	1.12	0.005	1.11	0.05
4	75	1.23	0.166	1.01	0.11	1.05	0.007	1.05	1.15	0.010	1.14	0.07
5	76	1.25	0.189	1.00	0.15	1.07	0.011	1.07	1.16	0.016	1.15	0.08

(c) *tsplib-middle*

m		**SOM**				**GENIUS-*quality***			**GENIUS-*fast***			
	$\#s$	LR	CQ	CER	s_{LR}	LR	CQ	CER	LR	CQ	CER	s_{LR}
1	88	1.05	-	-	0.01	1.04	-	-	1.06	-	-	0.01
2	89	1.10	0.085	1.01	0.08	1.03	0.001	1.03	1.06	0.001	1.06	0.04
3	89	1.17	0.137	1.01	0.12	1.04	0.002	1.04	1.08	0.002	1.08	0.04
4	89	1.28	0.185	1.04	0.15	1.05	0.003	1.05	1.13	0.003	1.13	0.05
5	90	1.36	0.233	1.04	0.18	1.06	0.003	1.06	1.15	0.004	1.15	0.06

(d) *tsplib-large*

Table B.10: Overall results for TSPLIB problems.



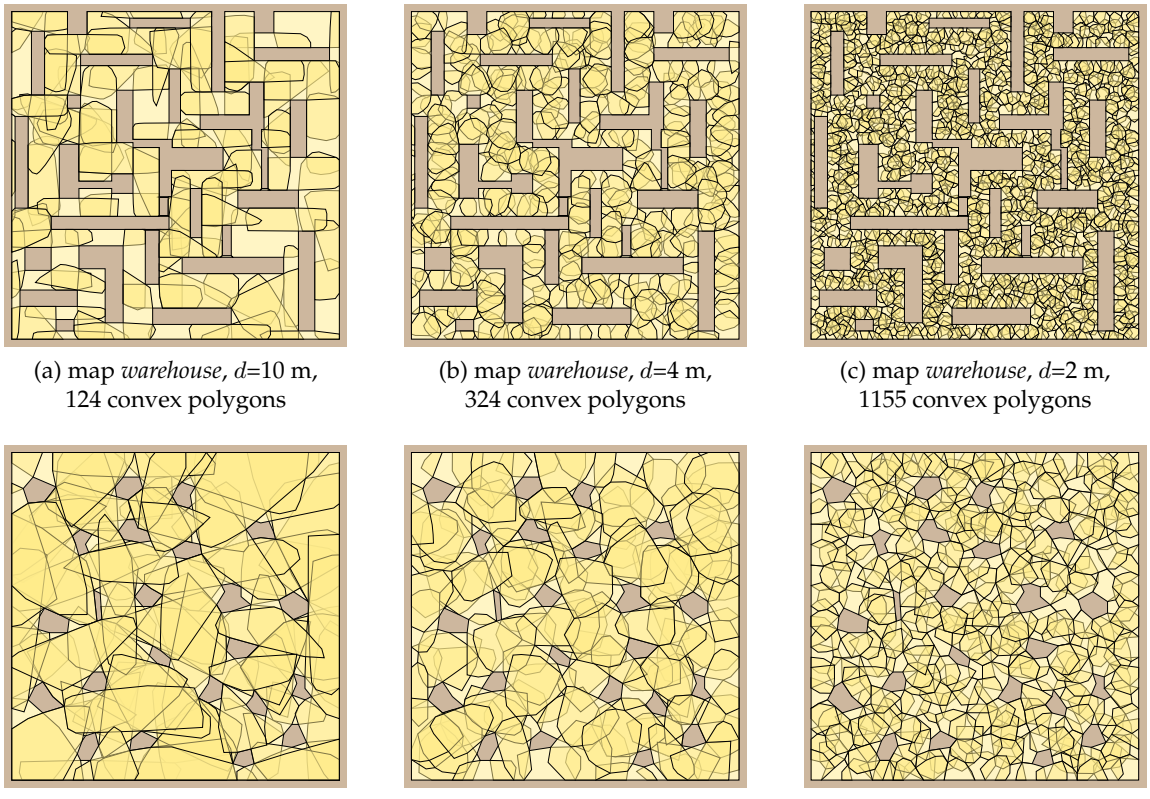
(a) SOM

(b) GENIUS

Figure B.3: Required computational time for TSPLIB problems.

Appendix C

Algorithm Experiments – WRP/MWRP



(a) map *warehouse*, d=10 m, 124 convex polygons

(b) map *warehouse*, d=4 m, 324 convex polygons

(c) map *warehouse*, d=2 m, 1155 convex polygons

(d) map *potholes*, d=10 m, 94 convex polygons

(e) map *potholes*, d=4 m, 132 convex polygons

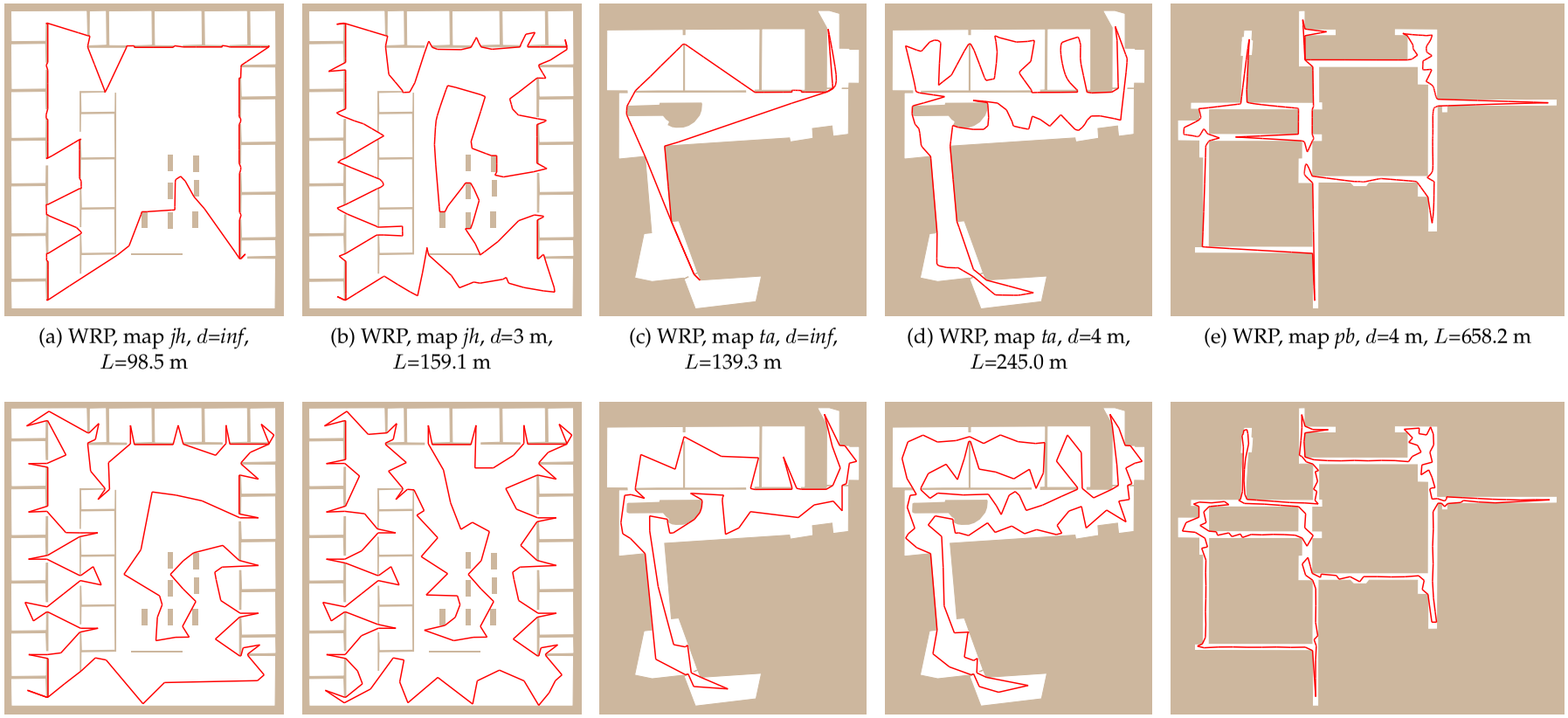
(f) map *potholes*, d=2 m, 357 convex polygons

Figure C.1: Examples of found convex covers for the visibility range d, a triangular mesh with 8210 triangles is used for the map *warehouse* and a mesh with 2429 triangles for the map *potholes*.

Map	d [m]	N_C	N_T	N_V	AGP+TSP		SOM - WRP					SOM - AGP+TSP			
					N_G	L_{ref} [m]	M	L [m]	PDM	PDB	T [s]	M	PDM	PDB	T [s]
jh	inf	100	872	576	77	193.3	88	103.5	-46.48	-49.02	6.3	192	1.47	0.73	5.5
jh	10.0	108	872	576	78	194.6	88	106.7	-45.16	-49.74	6.4	195	2.04	0.98	5.6
jh	5.0	130	872	576	85	204.3	88	114.8	-43.80	-47.89	7.1	212	1.93	0.82	6.9
jh	4.0	169	872	576	89	207.9	175	148.2	-28.69	-36.66	18.1	222	1.73	0.44	7.5
jh	3.0	258	931	608	100	215.5	187	164.8	-23.52	-26.19	32.5	250	2.51	0.61	9.4
jh	2.0	480	1 904	1 183	180	295.5	381	259.4	-12.19	-14.69	132.8	450	3.22	1.61	32.0
jh	1.5	852	2 272	1 392	282	359.0	682	337.5	-5.98	-7.53	474.4	705	3.37	2.78	77.6
jh	1.0	1 800	3 401	1 988	563	485.0	1 701	490.0	1.02	-0.16	3 277.1	1 408	4.00	3.12	336.8
ta	inf	46	1 014	638	46	215.6	102	140.7	-34.73	-35.39	1.2	115	2.24	0.43	1.7
ta	10.0	70	1 014	638	47	216.9	102	145.8	-32.78	-33.11	1.8	118	2.91	0.33	1.7
ta	5.0	152	1 014	638	70	256.8	102	215.1	-16.25	-18.66	6.4	175	1.17	0.40	3.8
ta	4.0	209	1 014	638	93	291.3	203	251.9	-13.54	-15.91	20.9	232	2.30	1.77	6.9
ta	3.0	357	1 252	776	138	335.6	376	297.1	-11.48	-13.36	73.0	345	1.51	1.04	15.2
ta	2.0	757	1 944	1 151	255	427.0	778	410.6	-3.84	-5.47	392.2	638	3.73	3.08	53.0
ta	1.5	1 320	3 117	1 788	432	538.5	1 247	528.4	-1.87	-2.81	1 232.2	1 080	5.17	4.59	161.8
ta	1.0	2 955	7 044	3 849	934	774.3	3 522	782.4	1.05	-0.35	8 587.4	2 335	6.15	5.86	816.2
pb	inf	52	2 403	1 630	50	554.9	241	434.2	-21.75	-23.86	6.2	125	4.60	0.17	1.9
pb	10.0	111	2 403	1 630	76	615.9	241	532.9	-13.48	-15.20	12.1	190	0.71	0.27	4.0
pb	5.0	262	2 403	1 630	134	687.2	241	643.2	-6.40	-9.60	30.3	335	0.95	0.25	12.4
pb	4.0	373	2 403	1 630	165	721.9	481	671.5	-6.98	-8.82	92.3	412	1.28	0.64	19.2
pb	3.0	714	3 078	2 018	244	781.6	616	729.8	-6.63	-7.52	244.0	610	2.32	0.58	42.1
pb	2.0	1 564	4 692	2 955	473	919.0	1 408	873.4	-4.96	-5.84	1 439.8	1 182	2.53	1.26	166.3
pb	1.5	2 787	7 144	4 319	870	1 158.2	2 858	1 127.4	-2.66	-3.33	6 181.5	2 175	2.84	1.98	601.7
pb	1.0	6 188	14 462	8 250	1 845	1 606.6*	5 785	1 602.8	-0.23	-1.41	34 090.1	4 612	4.14	3.22	2 785.0

Table C.1: WRP results, d - visibility range, N_C - size of the convex cover, N_T, N_V - the number of triangles, resp. vertices, of the triangular mesh, N_G - the number of guards, M - the number of neurons, T - the average value of the required computational time.

*Due to high number of guards the solution has been found by the Chained Lin-Kernighan heuristic [17].



(a) WRP, map *jh*, d=*inf*, L=98.5 m

(b) WRP, map *jh*, d=3 m, L=159.1 m

(c) WRP, map *ta*, d=*inf*, L=139.3 m

(d) WRP, map *ta*, d=4 m, L=245.0 m

(e) WRP, map *pb*, d=4 m, L=658.2 m

(f) AGP+TSP, map *jh*, d=*inf*, L=193.3 m

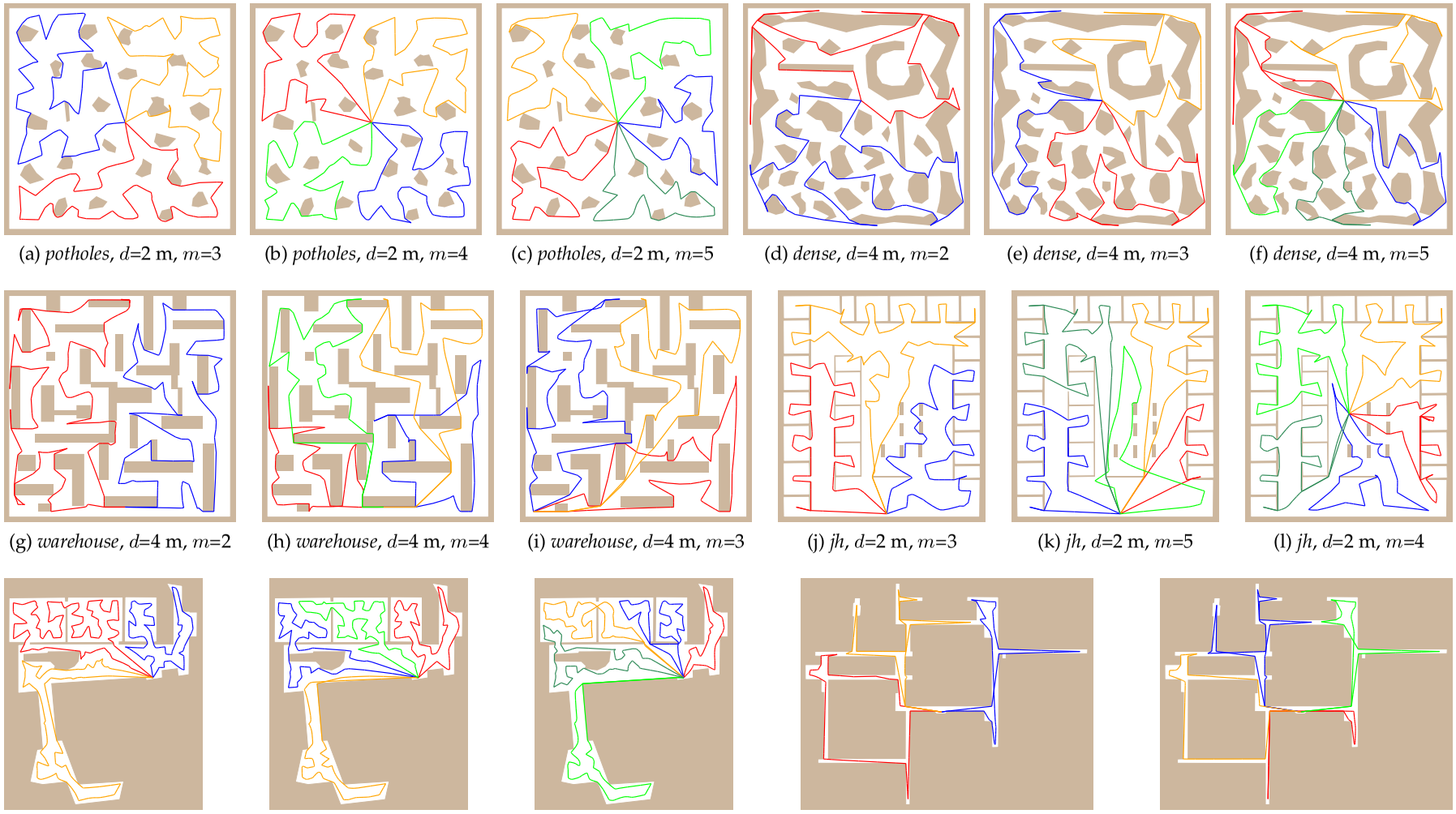
(g) AGP+TSP, map *jh*, d=3 m, L=215.5 m

(h) AGP+TSP, map *ta*, d=*inf*, L=215.6 m

(i) AGP+TSP, map *ta*, d=4 m, L=291.3 m

(j) AGP+TSP, map *pb*, d=4 m, L=721.9 m

Figure C.2: Selected the best solutions found by the proposed WRP algorithm and reference solutions for the visibility distance d, length of the route is denoted as L.

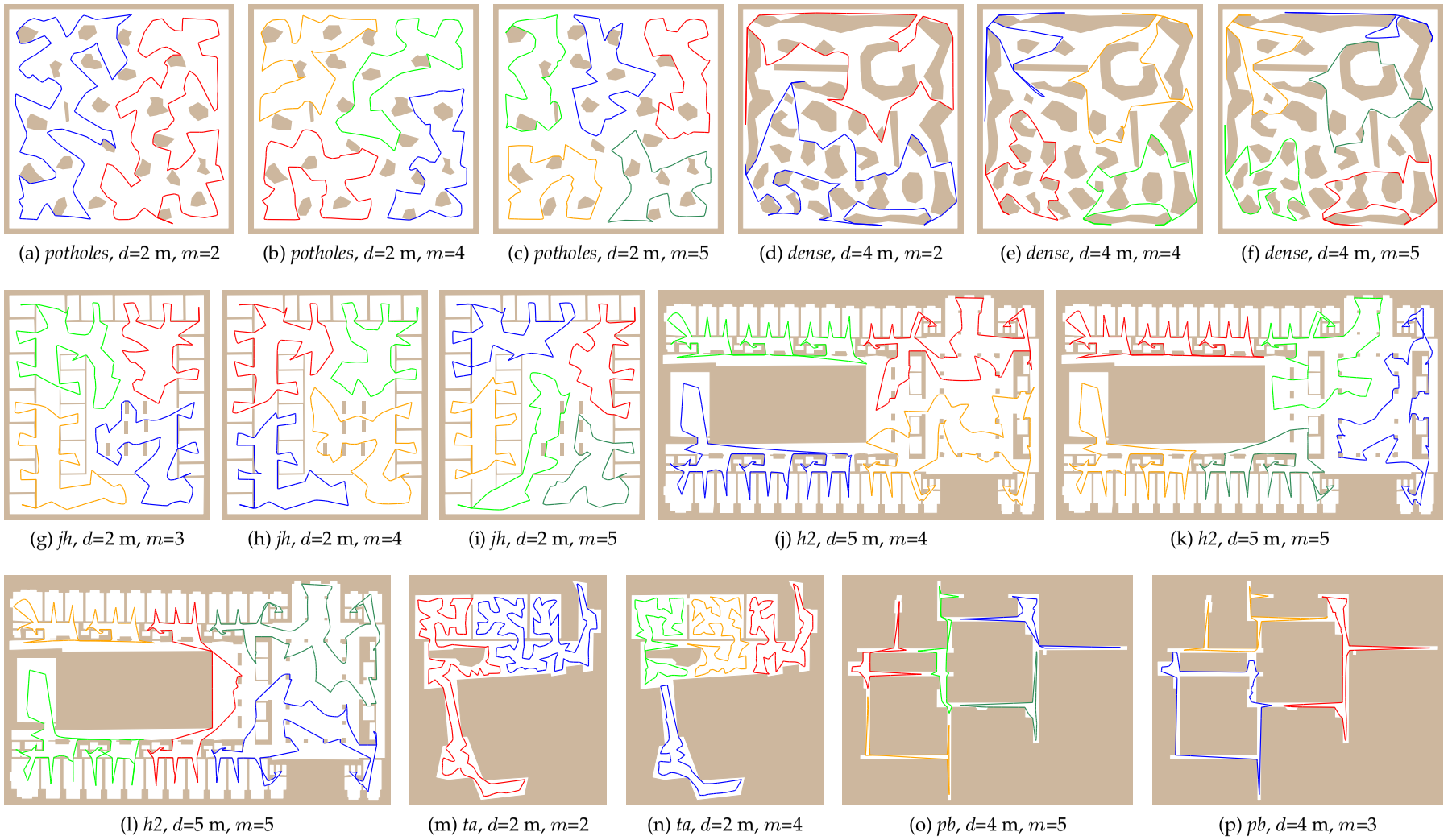


(a) *potholes*, d=2 m, m=3 (b) *potholes*, d=2 m, m=4 (c) *potholes*, d=2 m, m=5 (d) *dense*, d=4 m, m=2 (e) *dense*, d=4 m, m=3 (f) *dense*, d=4 m, m=5

(g) *warehouse*, d=4 m, m=2 (h) *warehouse*, d=4 m, m=4 (i) *warehouse*, d=4 m, m=3 (j) *jh*, d=2 m, m=3 (k) *jh*, d=2 m, m=5 (l) *jh*, d=2 m, m=4

(m) *ta*, d=2 m, m=3 (n) *ta*, d=2 m, m=4 (o) *ta*, d=2 m, m=5 (p) *pb*, d=4 m, m=3 (q) *pb*, d=4 m, m=4

Figure C.3: Selected solutions of the MWRP depot variant for m watchmen and the visibility range d.



(a) *potholes*, d=2 m, m=2 (b) *potholes*, d=2 m, m=4 (c) *potholes*, d=2 m, m=5 (d) *dense*, d=4 m, m=2 (e) *dense*, d=4 m, m=4 (f) *dense*, d=4 m, m=5

(g) *jh*, d=2 m, m=3 (h) *jh*, d=2 m, m=4 (i) *jh*, d=2 m, m=5 (j) *h2*, d=5 m, m=4 (k) *h2*, d=5 m, m=5

(l) *h2*, d=5 m, m=5 (m) *ta*, d=2 m, m=2 (n) *ta*, d=2 m, m=4 (o) *pb*, d=4 m, m=5 (p) *pb*, d=4 m, m=3

Figure C.4: Selected solutions of the MWRP without depot (patrolling routes) for m watchmen and the visibility range d.

Bibliography

[1] General-purpose computation on graphics hardware. http://gpgpu.org.

[2] LMS 200 Technical Specification. URL http://www.sick.com. SICK.

[3] Mobile Robots Inc. - P3AT - The High Performance All-Terrain Robot. http://www.activrobots.com/ROBOTS/p2at.html.

[4] Safety Laser Scanner S300 - Product Information. URL http://www.sick.com. SICK.

[5] Scanning laser ranger finder, utm-30lx/ln specification. Hokuyo Automatic Co., LTD.

[6] CGAL - Computational Geometry Algorithms Library. http://www.cgal.org, 2004.

[7] Product Specification, All Directional Camera Module, RPU-C2521 RPU-C3522, 2004. Sony Corporation.

[8] JTS Topology Suite. http://www.vividsolutions.com/jts/jtshome.htm, 2005. version 1.5.

[9] Diablo Caffe JDK 1.6.0-7. http://www.freebsdfoundation.org/downloads/java.shtml, 2009.

[10] GEOS - Geometry Engine, Open Source. http://trac.osgeo.org/geos, 2009. version 3.1.0.

[11] E. Acar, H. Choset, and J. Y. Lee. Sensor-based coverage with extended range detectors. *Robotics, IEEE Transactions on*, 22 (1):189–198, Feb. 2006.

[12] A. Aggarwal, D. Coppersmith, S. Khanna, R. Motwani, and B. Schieber. The angular-metric traveling salesman problem. In *SODA '97: Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, pages 221–229, New Orleans, Louisiana, USA, 1997. Society for Industrial and Applied Mathematics.

[13] Y. Amit, J. S. B. Mitchell, and E. Packer. Locating Guards for Visibility Coverage of Polygons. In *ALENEX*. SIAM, 2007.

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

[15] D. Applegate, W. Cook, S. Dash, and A. Rohe. Solution of a Min-Max Vehicle Routing Problem. *INFORMS J. on Computing*, 14(2):132–143, 2002.

[16] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Concorde TSP Solver. http://www.tsp.gatech.edu/concorde.html, 2003.

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

[18] T. Arai, E. Pagello, and L. E. Parker. Guest editorial advances in multirobot systems. *Robotics and Automation, IEEE Transactions on*, 18(5):655–661, October 2002.

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

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

[21] E. M. Arkin, M. A. Bender, E. D. Demaine, S. P. Fekete, J. S. B. Mitchell, and S. Sethia. Optimal covering tours with turn costs. In *SODA '01: Proceedings of the*

twelfth annual ACM-SIAM symposium on Discrete algorithms, pages 138–147, Washington, D.C., USA, 2001. Society for Industrial and Applied Mathematics.

[22] E. M. Arkin, J. S. B. Mitchell, and C. D. Piatko. Minimum-link watchman tours. *Information Processing Letters*, 86(4):203–207, 2003.

[23] D. J. Austin and B. J. McCarragher. Geometric constraint identification and mapping for mobile robots. *Robotics and Autonomous Systems*, 35(2):59–76, 2001.

[24] I. Averbakh and O. Berman. (p-1)/(p+1)-approximate algorithms for p-traveling salemen problems on a tree with minmax objective. *Discrete Appl. Math.*, 75(3):201–216, 1997.

[25] I. Averbakh and O. Berman. Minmax *p*-Traveling Salesmen Location Problems on a Tree. *Annals OR*, 110(1-4):55–68, 2002.

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

[27] J. Barraquand, B. Langlois, and J.-C. Latombe. Numerical potential field techniques for robot path planning. In *Advanced Robotics, 1991. 'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, volume 2, pages 1012–1017, Jun 1991.

[28] T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34 (3):209–219, 2006.

[29] M. Bellmore and S. Hong. Transformation of Multisalesman Problem to the Standard Traveling Salesman Problem. *Journal of the ACM*, 21(3):500–504, 1974.

[30] S. Bespamyatnikh. An $o(n \log n)$ algorithm for the Zoo-keeper's problem. *Computational Geometry: Theory and Applications*, 24(2):63–74, 2002.

[31] P. Bhattacharya and M. Gavrilova. Roadmap-Based Path Planning - Using the Voronoi Diagram for a Clearance-Based Shortest Path. *Robotics & Automation Magazine, IEEE*, 15(2):58–66, 2008.

[32] I. Bjorling-Sachs and D. Souvaine. A Tight Bound for Guarding Polygons with Holes. Technical report, Rutgers University, 1991. Report LCSR-TR-165.

[33] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

[34] J. Borenstein and Y. Koren. Histogramic in-motion mapping for mobile robot obstacle avoidance. *IEEE Journal of Robotics and Automation*, 7(4):535–539, 1991.

[35] M. C. Bosse. *ATLAS: a framework for large scale automated mapping and localization*. PhD thesis, Massachusetts Institute of Technology, 2004.

[36] A. Bottino and A. Laurentini. Optimal Positioning of Sensors in 2D. In *CIARP*, pages 53–58, 2004.

[37] A. Bottino and A. Laurentini. Optimal Positioning of Sensors in 3D. In *CIARP*, pages 804–812, 2005.

[38] J. Bruce and M. Veloso. Real-Time Randomized Path Planning for Robot Navigation. In *IEEE/RSJ International Conference on Intelligent Robots and System*, volume 3, pages 2383–2388, 2002.

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

[40] M. Budinich and B. Rosario. A neural network for the travelling salesman problem with a well behaved energy function. In *MANNA '95: Proceedings of the first international conference on Mathematics of neural networks : models, algorithms and applications*, pages 134–139, Norwell, MA, USA, 1997. Kluwer Academic Publishers.

[41] V. Bugera. Properties of No-Depot Min-Max 2-Traveling-Salesmen Problem. In S. Butenko, R. Murphey, and P. M. Pardalos, editors, *Recent Developments in Cooperative Control and Optimization*. Kluwer Academic Publishers, Norwell, MA, USA, 2004.

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

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

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

[45] S. Carlsson and B. J. Nilsson. Computing Vision Points in Polygons. *Algorithmica*, 24(1):50–75, 1999.

[46] S. Carlsson, B. J. Nilsson, and S. C. Ntafos. Optimum Guard Covers and m-Watchmen Routes for Restricted Polygons. *International Journal of Computational Geometry and Applications*, 3(1):85–105, 1993.

[47] P. Cheng. A Short Survey on Pursuit-Evasion Games, 2003. Lecture Notes.

[48] Y.-J. Chiang and J. S. B. Mitchell. Two-point Euclidean shortest path queries in the plane. In *SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 215–224, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.

[49] W.-P. Chin and S. Ntafos. Optimum watchman routes. In *SCG '86: Proceedings of the second annual symposium on Computational geometry*, pages 24–33, Yorktown Heights, NY, USA, 1986. ACM Press.

[50] W.-P. Chin and S. Ntafos. The zookeeper route problem. *Information Sciences: an International Journal*, 63(3):245–259, 1992.

[51] H. Choset and P. Pignon. Coverage path planning: The boustrophedon cellular decomposition. In *International Conference on Field and Service Robotics*, Canberra, Australia, 1997.

[52] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents)*. The MIT Press, June 2005.

[53] V. Chvátal. A Combinatorial Theorem in Plane Geometry. *Journal of Combinatorial Theory (B)*, 18:39–41, 1975.

[54] G. Clarke and J. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4): 568–581, 1964.

[55] J. Clossey, G. Laporte, and P. Soriano. Solving arc routing problems with turn penalties. *Journal of the Operational Research Society*, 52(4):433–439, April 2001.

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

[57] C. I. Connolly and R. A. Grupen. Applications of Harmonic Functions to Robotics. Technical report, University of Massachusetts, Amherst, MA, USA, 1992.

[58] C. I. Connolly, J. B. Burns, and R. Weiss. Path Planning Using Laplace's Equation. In *In Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, pages 2102–2106, 1990.

[59] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd revised edition edition, September 2001.

[60] M. Cottrell, J.-C. Fort, and G. Pagès. Theoretical Aspects of the SOM Algorithm. *Neurocomputing*, 21:119–138, 1998.

[61] M. C. Couto, C. C. S. de Souza, and P. J. de Rezende. An Exact and Efficient Algorithm for the Orthogonal Art Gallery Problem. In *SIBGRAPI '07: Proceedings of the XX Brazilian Symposium on Computer Graphics and Image Processing*, pages 87–94, Washington, DC, USA, 2007. IEEE Computer Society.

[62] M. C. Couto, P. J. de Rezende, and C. C. de Souza. An IP solution to the art gallery problem. In *SCG '09: Proceedings of the 25th annual symposium on Computational geometry*, pages 88–89, New York, NY, USA, 2009. ACM.

[63] J.-C. Créput and A. Koukam. The memetic self-organizing map approach to

the vehicle routing problem. *Soft Comput.*, 12(11):1125–1141, 2008.

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

[65] J. C. Culberson and R. A. Reckhow. Covering polygons is hard. *Journal of Algorithms*, 17(1):2–44, 1994.

[66] J. Czyzowicz, P. Egyed, H. Everett, D. Rappaport, T. Shermer, D. Souvaine, G. Toussaint, and J. Urrutia. The Aquarium Keeper's Problem. In *SODA '91: Proceedings of the second annual ACM-SIAM symposium on Discrete algorithms*, pages 459–464, San Francisco, California, USA, 1991. Society for Industrial and Applied Mathematics.

[67] J. Czyzowicz, E. Rivera-Campo, N. Santoro, J. Urrutia, and J. Zaks. Guarding rectangular art galleries. *Discrete Applied Mathematics*, 50(2):149–157, 1994.

[68] R. Daily and D. Bevly. Harmonic potential field path planning for high speed vehicles. In *American Control Conference, 2008*, pages 4609–4614, June 2008.

[69] T. Danner and L. E. Kavraki. Randomized Planning for Short Inspection Paths. In *Proceedings of The IEEE International Conference on Robotics and Automation (ICRA)*, pages 971–976, San Fransisco, CA, April 2000. IEEE Press.

[70] E. R. Davies. *Machine Vision, Third Edition: Theory, Algorithms, Practicalities (Signal Processing and its Applications)*. Morgan Kaufmann, 3 edition, January 2005.

[71] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, Berlin, 2nd ed. edition, 2000.

[72] O. Devillers, S. Pion, M. Teillaud, and P. Prisme. Walking in a Triangulation. *Internat. J. Found. Comput. Sci*, 13:106–114, 2001.

[73] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26(1):29–41, 1996.

[74] M. Dror, A. Efrat, A. Lubiw, and J. S. B. Mitchell. Touring a sequence of polygons. In *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 473–482, San Diego, CA, USA, 2003. ACM Press.

[75] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, 2000.

[76] A. Dumitrescu and J. S. B. Mitchell. Approximation algorithms for TSP with neighborhoods in the plane. In *SODA '01: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 38–46, Washington, D.C., USA, 2001. Society for Industrial and Applied Mathematics.

[77] R. Durbin and D. Willshaw. An analogue approach to the travelling salesman problem using an elastic net method. *Nature*, 326(16):689–691, April 1987.

[78] E. P. e Silva Jr., P. M. Engel, M. Trevisan, and M. A. P. Idiart. Exploration method using harmonic functions. *Robotics and Autonomous Systems*, 40(1):25–42, 2002.

[79] K. Easton and J. Burdick. A Coverage Algorithm for Multi-robot Boundary Inspection. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 727–734, April 2005.

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

[81] S. Eidenbenz, C. Stamm, and P. Widmayer. Inapproximability of some art gallery problems. In *In Proceedings of the 10th Canadian Conference on Computational Geometry (CCCG'98)*, Montréal, Québec, Canada, 1998.

[82] U. M. Erdem and S. Sclaroff. Automated Placement of Cameras in a Floorplan to Satisfy Task-Specific Constraints. Technical report, Boston University, 2003.

[83] U. M. Erdem and S. Sclaroff. Automated camera layout to satisfy task-specific and floor plan-specific coverage

requirements. *Computer Vision and Image Understanding*, 103(3):156–169, 2006.

[84] J. Faigl. Multi-Robots Coverage and Exploration Path Planning : Current State of the Art. Research Report GLR 175/04, Gerstner Laboratory, Czech Technical University in Prague, Prague, 2004.

[85] J. Faigl and M. Kulich. Sensing locations positioning for multi-robot inspection planning. In *DIS '06: Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)*, pages 79–84, Washington, DC, USA, 2006. IEEE Computer Society.

[86] J. Faigl, G. Klančar, D. Matko, and M. Kulich. Path planning for multi-robot inspection task considering acceleration limits. In *Proceedings of the fourteenth International Electrotechnical and Computer Science Conference ERK 2005*, 2005.

[87] J. Faigl, T. Krajník, M. Saska, K. Košnar, and J. Chudoba. Fira MiroSot - G-Bots team description. Fira EuroCup Robot Soccer Workshop, 2005.

[88] J. Faigl, M. Kulich, and L. Přeučil. Multiple traveling salesmen problem with hierarchy of cities in inspection task with limited visibility. In *5th Workshop on Self-Organizing Maps*, pages 91–98, 2005.

[89] G. Faria, R. Romero, E. Prestes, and M. Idiart. Comparing harmonic functions and potential fields in the trajectory control of mobile robots. In *Robotics, Automation and Mechatronics, 2004 IEEE Conference on*, volume 2, pages 762–767, Dec. 2004.

[90] S. P. Fekete. *Geometry and the Travelling Salesman Problem*. Ph.D. thesis, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, 1992.

[91] S. P. Fekete and G. J. Woeginger. Angle-Restricted Tours in the Plane. *Computational Geometry*, 8(225):195–218, 1997.

[92] D. Ferguson and A. T. Stentz. Anytime RRTs. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '06)*, pages 5369–5375, October 2006.

[93] S. Fisk. A short proof of Chvatal's watchman theorem. *Journal of Combinatorial Theory Series B*, 24:374+, 1978.

[94] E. Fogel and D. Halperin. Exact and efficient construction of Minkowski sums of convex polyhedra with applications. In *Proc. 8th Wrkshp. Alg. Eng. Exper. (Alenex'06)*, pages 3–15, 2006.

[95] E. Fogel, R. Wein, and D. Halperin. Code Flexibility and Program Efficiency by Genericity: Improving CGAL's Arrangements. In *ESA*, pages 664–676, 2004.

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

[97] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Trans. Math. Softw.*, 33(2):13, 2007.

[98] P. M. França, M. Gendreau, G. Laporte, and F. M. Müller. The m-Traveling Salesman Problem with Minmax Objective. *Transportation Science*, 29(3):267–275, 1995.

[99] M. L. Fredman, D. S. Johnson, L. A. Mc Geoch, and G. Ostheimer. Data structures for traveling salesmen. In *SODA '93: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 145–154, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

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

[101] T. Furukawa, K. Tokunaga, K. Morishita, and S. Yasui. Modular network SOM (mnSOM): from vector space to function space. In *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, volume 3, pages 1581–1586, July-4 Aug. 2005.

[102] A. Ganguli, J. Cortés, and F. Bullo. Distributed deployment of asynchronous guards in art galleries. In *American Control Conference*, 2006.

[103] S. Garrido, L. Moreno, D. Blanco, and F. M. Monar. Robotic Motion Using Harmonic Functions and Finite Elements. *Journal of Intelligent and Robotic Systems*, 2009.

[104] R. Geraerts and M. H. Overmars. Creating High-quality Paths for Motion Planning. *Int. J. Rob. Res.*, 26(8):845–863, 2007.

[105] R. Geraerts and M. H. Overmars. The corridor map method: a general framework for real-time high-quality path planning: Research Articles. *Comput. Animat. Virtual Worlds*, 18(2):107–119, 2007.

[106] H. Ghaziri and I. H. Osman. A neural network algorithm for the traveling salesman problem with backhauls. *Computers and Industrial Engineering*, 44(2):267–281, 2003.

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

[108] H. González-Baños and J.-C. Latombe. Planning Robot Motions for Range-Image Acquisition and Automatic 3D Model Construction. In *AAAI Fall Symposium*, 1998.

[109] J. E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry*. CRC Press LLC, Boca Raton, FL, 2004.

[110] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: a hierarchical structure for rapid interference detection. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180, New York, NY, USA, 1996. ACM.

[111] T. Granlund. GNU MP: The GNU Multiple Precision Arithmetic Library, 2004. URL `http://gmplib.org`.

[112] Gregory Dudek and Souad Hadjres and Paul Freedman. Using Local Information in a Non-local Way for Mapping Graph-like Worlds. In *Proceedings of International Joint Conference of Artificial Intelligence*, Chambery, France, 1993.

[113] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. A Visibility-Based Pursuit-Evasion Problem. *International Journal of Computational Geometry and Applications*, 9:471+, 1999.

[114] P. Hachenberger, L. Kettner, and K. Mehlhorn. Boolean operations on 3D selective Nef complexes: Data structure, algorithms, optimized implementation and experiments. *Comput. Geom. Theory Appl.*, 38(1-2):64–99, 2007.

[115] M. Hachicha, M. J. Hodgson, G. Laportert, and F. Semet. Heuristics for the multi-vehicle covering tour problem. *Computers and Operations Research*, 27(1): 29–42, 2000.

[116] B. Hammer and T. Villmann. Mathematical Aspects of Neural Networks. In *European Symposium of Artificial Neural Networks 2003*, pages 59–72. D-side Publications, 2003.

[117] J. Hancock, E. Hoffman, R. Sullivan, D. Ingimarson, D. Langer, and M. Hebert. High-performance laser range scanner. In *SPIE Proceedings on Intelligent Transportation Systems*, 1997.

[118] K. Helsgaun. An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. *European Journal of Operational Research*, 126(1), 2000.

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

[120] J. P. Hespanha, H. J. Kim, and S. Sastry. Multiple-Agent Probabilistic Pursuit-Evasion Games. In *Proceedings of the 38th Conference on Decision and Control*, volume 3, pages 2432–2437, 1999.

[121] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel. The Polygon Exploration Problem. *SIAM Journal on Computing*, pages 577–600, 2001.

[122] R. Honsberger. *Mathematical Gems II*. Mathematical Association of America, 1979.

[123] J. J. Hopfield and D. W. Tank. "Neural" computation of decisions in optimization

problems. *Biological Cybernetics*, 52(3): 141–152, 1985.

[124] E. Hörster and R. Lienhart. On the optimal placement of multiple visual sensors. In *VSSN '06: Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, pages 111–120, New York, NY, USA, 2006. ACM.

[125] A. Howard. Multi-robot mapping using manifold representations. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 4, pages 4198–4203, May 2004.

[126] W. Huang. Optimal Line-sweep-based Decompositions for Coverage Algorithms. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, 2001.

[127] W. H. Huang and K. R. Beevers. Complete topological mapping with sparse sensing. Technical Report 6, Rensselaer Polytechnic Institute Department of Computer Science, March 2005.

[128] Y. Huang and K. Gupta. RRT-SLAM for motion planning with motion and map uncertainty for robot exploration. In *IROS*, pages 1077–1082, 2008.

[129] S. Hutchinson, R. Cromwell, and A. Kak. Applying Uncertainty Reasoning To Model Based Object Recognitio. *Computer Vision and Pattern Recognition*, 89: 541–548, 1989.

[130] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion with limited visibility. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1060–1069, New Orleans, Louisiana, 2004. Society for Industrial and Applied Mathematics.

[131] P. Jacobs and J. Canny. Planning smooth paths for mobile robots. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, volume 1, pages 2–7, May 1989.

[132] L. Jaillet, A. Yershova, S. M. LaValle, and T. Siméon. Adaptive Tuning of the Sampling Domain for Dynamic-Domain RRTs. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005*, pages 3856–3861, 2005.

[133] R. A. Jarvis. Robot path planning: complexity, flexibility and application scope. In *PCAR '06: Proceedings of the 2006 international symposium on Practical cognitive agents and robots*, pages 3–14, New York, NY, USA, 2006. ACM.

[134] L. Jiao and Z. Tang. A Visibility-based Algorithm for Multi-robot Boundary Coverage. *International Journal of Advanced Robotic Systems*, 5(1), March 2008.

[135] H.-D. Jin, K.-S. Leung, M.-L. Wong, and Z.-B. Xu. Designing an Expanded SOM for Traveling Salesman Problem by Genetic Algorithms. In *Proceedings of the 2000 Genetic and Evolutionary Computation Conference*, 2000.

[136] H.-D. Jin, K.-S. Leung, M.-L. Wong, and Z.-B. Xu. An Integrated Self-Organizing Map for the Traveling Salesman Problem. In *Advances in Neural Networks and Applications*, pages 235–240, 2001.

[137] H.-D. Jin, K.-S. Leung, M.-L. Wong, and Z.-B. Xu. An efficient self-organizing map designed by genetic algorithms for the traveling salesman problem. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 33(6):877–888, Dec. 2003.

[138] D. Johnson and L. A. McGeoch. The Traveling Salesman Problem: A Case Study. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley & Sons, New York, 1997.

[139] D. S. Johnson and L. A. McGeoch. Experimental Analysis of Heuristics for The STSP. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*, chapter 9, pages 369–443. Kluwer Academic Publishers, Boston, 2002.

[140] D. S. Johnson, G. Gutin, L. A. McGeoch, A. Yeo, W. Zhang, and A. Zverovich. Experimental Analysis of Heuristics for The ATSP. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*, chapter 10, pages

445–487. Kluwer Academic Publishers, Boston, 2002.

[141] M. Kalisiak. *Toward More Efficient Motion Planning with Differential Constraints*. PhD thesis, University of Toronto, 2007.

[142] M. Kalisiak and M. van de Panne. RRT-blossom: RRT with a Local Flood-fill Behavior. In *ICRA*, pages 1237–1242, 2006.

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

[144] G. Kamberova and M. Mintz. Robust multi-sensor fusion & decision theoretic approach. In *roceedings of the 1990 DARPA Image Understanding Workshop*, pages 867–873. Morgan Kaufmann Publishers, Inc., September 1990.

[145] S. Kapoor. Efficient computation of geodesic shortest paths. In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 770–779, New York, NY, USA, 1999. ACM.

[146] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic Roadmaps for Path Planning in High Dimensional Configuration Spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[147] G. Kazazakis and A. Argyros. Fast Positioning of Limited Visibility Guards for the Inspection of 2D Workspaces. In *Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS 2002)*, Lausanne, Switzerland, September 2002.

[148] S. Kernbach, L. Ricotti, J. Liedke, P. Corradi, and M. Rothermel. Study of Macroscopic Morphological Features of Symbiotic Robotic Organisms. In *IROS 2008 Workshop on Self-Reconfigurable Robots/Systems and Applications*, 2008.

[149] C. H. Kim and B. K. Kim. Minimum-Energy Translational Trajectory Generation for Differential-Driven Wheeled Mobile Robots. *Journal of Intelligent and Robotic Systems*, 49(4):367–383, 2007.

[150] J.-O. Kim and P. Khosla. Real-time obstacle avoidance using harmonic potential functions. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, volume 1, pages 790–796, April 1991.

[151] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, May 1983.

[152] R. Klein and Z. N. Elmar Langetepe. Abstract Voronoi diagrams revisited. *Comput. Geom. Theory Appl.*, 42(9):885–902, 2009.

[153] A. Klesh, P. Kabamba, and A. Girard. Path planning for cooperative time-optimal information collection. In *American Control Conference, 2008*, pages 1991–1996, June 2008.

[154] T. Kohonen. Self-Organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[155] F. A. Kolushev and A. A. Bogdanov. Multi-agent Optimal Path Planning for Mobile Robots in Environment with Obstacles. In *Ershov Memorial Conference*, pages 503–510, 1999.

[156] J. Kubalík and J. Faigl. Iterative prototype optimisation with evolved improvement steps. In *Proceedings of the 9th European Conference on Genetic Programming*, volume 3905 of *Lecture Notes in Computer Science*, pages 154–165, Budapest, Hungary, April 2006. Springer.

[157] J. Kubalík, J. Kléma, and M. Kulich. Application of Soft Computing Techniques to Rescue Operation Planning. In *Artificial Intelligence and Soft Computing - ICAISC 2004*, pages 897–902, Berlin, 2004. Springer.

[158] M. Kulich. *Localization and Creation of Environment Model in Mobile Robotics*. PhD thesis, Czech Technical University in Prague, 2003. in Czech.

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

[160] M. Kulich, J. Kout, and L. P. et al. PeLoTe - a Heterogeneous Telematic System for Cooperative Search and Rescue Missions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems 2004*, volume 1, Sendai, 2004.

[161] M. Kulich, J. Faigl, and L. Přeučil. Cooperative planning for heterogeneous teams in rescue operations. In *IEEE International Workshop on Safety, Security and Rescue Robotics*, 2005.

[162] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How. Motion planning for urban driving using RRT. In *IROS*, pages 1681–1686, 2008.

[163] M. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, and A. Kleywegt. Simple auctions with performance guarantees for multi-robot task allocation. In *Proc. Intl. Conf on Intelligent Robots and Systems (IROS 2004)*, volume 1, pages 698–705, Sept.-2 Oct. 2004.

[164] R. Lakämper, L. J. Latecki, X. Sun, and D. Wolter. Geometric Robot Mapping. In *DGCI*, pages 11–22, 2005.

[165] G. Laport and I. H. Osman. Routing problems: A bibliography. *Annals of Operations Research*, 61:227–262, 1995.

[166] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, June 1992.

[167] G. Laporte, M. Desrochers, and Y. Nobert. Two exact algorithms for the distance-constrained vehicle routing problem. *Networks*, 14(1):161–172, 1984.

[168] G. Laporte, Y. Nobert, and S. Taillefer. A Branch-and-Bound Algorithm for the Asymmetrical Distance Constrained Vehicle Routing Problem. *Mathematical Modelling*, 9(12):857–868, 1987.

[169] G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet. Classical and Modern Heuristics for the vehicle routing problem. *International Transaction in Operational Research*, 7:285–300, 2000.

[170] L. Latecki and R. Lakämper. Polygonal approximation of laser range data based on perceptual grouping and EM. In *Proc. Intl. Conf. on Robotics and Automation (ICRA)*, pages 790–796, May 2006.

[171] L. J. Latecki and R. Lakämper. Convexity rule for shape decomposition based on discrete contour evolution. *Comput. Vis. Image Underst.*, 73(3):441–454, 1999.

[172] D. Latimer IV, S. Srinivasa, V. L. Shue, S. Sonne, H. Choset, and A. Hurst. Towards sensor based coverage with robot teams. In *Proc. Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, May 2002.

[173] J.-C. Latombe. *Robot Motion Planning*. International Series in Engineering and Computer Science; Robotics: Vision, Manipulation and Sensors. Kluwer Academic Publishers, Boston, MA, U.S.A., 1991.

[174] A. Laurentini. Guarding the walls of an art gallery. *The Visual Computer*, 15(6): 265–278, 1999.

[175] S. LaValle, D. Lin, L. Guibas, J.-C. Latombe, and R. Motwani. Finding an unpredictable target in a workspace with obstacles. In *Proc. IEEE International Conference on Robotics and Automation*, pages 737–742, 1997.

[176] S. M. Lavalle. *Planning Algorithms*. Cambridge University Press, May 2006.

[177] S. M. Lavalle and J. J. K. Jr. Rapidly-Exploring Random Trees: Progress and Prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308, 2000.

[178] S. M. Lavalle and J. J. Kuffner. Randomized kinodynamic planning. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 1, pages 473–479, 1999.

[179] S. M. LaValle, B. H. Simov, and G. Slutzki. An algorithm for searching a polygonal region with a flashlight. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, pages 260–269, New York, NY, USA, 2000. ACM.

[180] P. Lefèvre, A. Prüß, and U. R. Zimmer. ALICE - Topographic Exploration, Cartography and Adaptive Navigation on

a Simple Mobile Robot. In *TSRPC '94*, Leeuwenhorst, The Netherlands, 1994.

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

[182] S. Lin and B. W. Kernighan. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, 21(2):498–516, 1973.

[183] A. Lingas. The Power of Non-Rectilinear Holes. In *Proceedings of the 9th Colloquium on Automata, Languages and Programming*, pages 369–383. Springer-Verlag, 1982.

[184] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, 22(10):560–570, 1979.

[185] J. Mačák. Multi-Robot Cooperative Inspection Task. Master's thesis, Czech Technical University in Prague, Czech Republic, 2009. in Czech.

[186] W. Malik, S. Rathinam, S. Darbha, and D. Jeffcoat. Combinatorial Motion Planning of Multiple Vehicle Systems. In *Decision and Control, 2006 45th IEEE Conference on*, pages 5299–5304, Dec. 2006.

[187] T. Martinetz and K. Schulten. Topology representing networks. *Neural Networks*, 7(3):507 – 522, 1994.

[188] M. S. Marzouqi and R. A. Jarvis. New visibility-based path-planning approach for covert robotic navigation. *Robotica*, 24 (6):759–773, 2006.

[189] E. Masehian and M. R. Amin-Naseri. A voronoi diagram-visibility graph-potential field compound algorithm for robot path planning. *J. Robot. Syst.*, 21(6): 275–300, 2004.

[190] M. J. Mataric. Issues and approaches in the design of collective autonomous agents. *Robotics and Autonomous Systems*, 16:321–331, December 1995.

[191] N. Melchior and R. Simmons. Particle RRT for Path Planning with Uncertainty. In *2007 IEEE International Conference on Robotics and Automation*, pages 1617–1624, April 2007.

[192] T. S. Michael and V. Pinciu. Art gallery theorems for guarded guards. *Computational Geometry: Theory and Applications*, 26 (3):247–258, 2003.

[193] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs (2nd, extended ed.)*. Springer-Verlag New York, Inc., 1994.

[194] J. S. B. Mitchell. A new algorithm for shortest paths among obstacles in the plane. *Annals of Mathematics and Artificial Intelligence*, 3(1), March 1991.

[195] J. S. B. Mitchell. Shortest paths among obstacles in the plane. In *SCG '93: Proceedings of the ninth annual symposium on Computational geometry*, pages 308–317, San Diego, California, USA, 1993. ACM Press.

[196] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24(11):1097–1100, 1997.

[197] R. Motwani, A. Raghunathan, and H. Saran. Covering orthogonal polygons with star polygons: the perfect graph approach. In *SCG '88: Proceedings of the fourth annual symposium on Computational geometry*, pages 211–223, Urbana-Champaign, Illinois, USA, 1988. ACM Press.

[198] D. M. Mount and S. Arya. ANN: A Library for Approximate Nearest Neighbor Searching. `http://www.cs.umd.edu/~mount/ANN`, 2006. Version 1.1.1.

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

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

[201] N. J. Nilsson. A Mobile Automaton: An Application of Artificial Intelligence Techniques. In *IJCAI*, pages 509–520, 1969.

[202] S. Ntafos. Watchman routes under limited visibility. *Computational Geometry: Theory and Applications*, 1(3):149–170, 1992.

[203] A. Nüchter. *3D Robotic Mapping. The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom*, volume 52 of *Springer Tracts in Advanced Robotics*. Springer Publishing Company, Incorporated, 2009.

[204] J. O'Rourke. Galleries need fewer mobile guards: a variation on Chvátal's theorem. *Geometriae Dedicata*, 14:273–283, 1983.

[205] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, August 1987.

[206] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1998.

[207] Y. Osais, M. St-Hilaire, and F. R. Yu. Directional Sensor Placement with Optimal Sensing Range, Field of View and Orientation. *Wireless and Mobile Computing, Networking and Communication, IEEE International Conference on*, 0:19–24, 2008.

[208] C. Osterman, C. Rego, and D. Gamboa. On the Performance of Data Structures for the Traveling Salesman Problem. In J. Cha, R. Jardim-Goncalves, and A. Steiger-Garcao, editors, *Proceedings of the 10th International Conference on Concurrent Engineering: Research and Applications*, pages 365–371, Netherlands, 2003. Balkema Publishers.

[209] T. Otani and M. Koshino. Applying a path planner based on RRT to cooperative multirobot box-pushing. *Artificial Life and Robotics*, 13(2):418–422, March 2009.

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

[211] E. Packer. *Robust Geometric Computing and Optimal Visibility Coverage*. PhD thesis, Stony Brook University, New York, 2008.

[212] L. E. Parker. Current State of the Art in Distributed Autnomous Mobile Robotics. In *Distributed Autonomous Robotic Systems 4 (DARS)*, pages 3–14, 2000.

[213] J. Pedroso. Niche search: an application in vehicle routing. In *IEEE World Congress on Computational Intelligence*, 1998.

[214] A. Plebe. An Effective Traveling Salesman Problem Solver Based on Self-Organizing Map. In *ICANN '02: Proceedings of the International Conference on Artificial Neural Networks*, pages 908–913, London, UK, 2002. Springer-Verlag.

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

[216] M. Pöllä, T. Honkela, and T. Kohonen. Bibliography of Self-Organizing Map (SOM) Papers. unpublished manuscript, 2006.

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

[218] A. Ranganathan and S. Koenig. PDRRTs: Integrating Graph-Based and Cell-Based Planning. In *Intelligent Robots and Systems*, volume 3, pages 2799–2806, 2004.

[219] J. A. Reeds and L. A. Sheep. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(4):367–393, 1990.

[220] J. H. Reif. Complexity of the mover's problem and generalizations. In *SFCS '79: Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 421–427, Washington, DC, USA, 1979. IEEE Computer Society.

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

[222] S. K. Sachindra, S. Kapoor, S. N. Maheshwari, and J. S. B. Mitchell. An Efficient Algorithm for Euclidean Shortest Paths Among Polygonal Obstacles in the Plane. *Discrete & Computational Geometry*, 18(4): 172–182, 1997.

[223] A. A. Sadeghi. Convergence in distribution of the multidimensional Kohonen algorithm. *Journ. of App. Prob*, 38:200–1, 2001.

[224] S. Safra and O. Schwartz. On the complexity of approximating tsp with neighborhoods and related problems. *Computational Complexity*, 14(4):281–307, 2006.

[225] M. Saha, G. Sánchez-Ante, and J.-C. Latombe. Planning multigoal tours for robot arms. In *Proceedings of IEEE International Conference on Robotics and Automation, ICRA '03*, volume 3, pages 3797–3803, 2003.

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

[227] H. Samet. The Quadtree and Related Hierarchical Data Structures. *ACM Computing Surveys (CSUR)*, 16(2):187–260, 1984.

[228] A. Sarmiento, R. Murrieta-Cid, and S. Hutchinson. Planning Expected-time Optimal Paths for Searching Known Environments. In *Proccedings IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2004*, Sendai, Japan., 2004.

[229] A. Sarmiento, R. Murrieta-Cid, and S. Hutchinson. A Multi-robot Strategy for Rapidly Searching a Polygonal Environment. In *Proceedings 9th Ibero-American Conference on Artificial Intelligence*, Puebla, Mexico, 2004. Best paper award.

[230] S. Schirra. How Reliable Are Practical Point-in-Polygon Strategies? In *ESA '08: Proceedings of the 16th annual European symposium on Algorithms*, pages 744–755, Berlin, Heidelberg, 2008. Springer-Verlag.

[231] M. Schwardt and J. Dethloff. Solving a continuous location-routing problem by use of a self-organizing map. *International Journal of Physical Distribution & Logistics Management*, 35(6):390–408, 2005.

[232] W. R. Scott, G. Roth, and J.-F. Rivest. View planning for automated three-dimensional object reconstruction and inspection. *ACM Comput. Surv.*, 35(1):64–96, 2003.

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

[234] K. T. Seow and M. Pasquier. Vehicle route-sequence planning using temporal logic. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 16(1): 31–38, 2002.

[235] T. Shermer. On recognizing unions of two convex polygons and related problems. *Pattern Recognition Letters*, 14(9):737–745, September 1993.

[236] T. C. Shermer. Recent Results in Art Galleries. In *Proccedings of the IEEE*, volume 80, pages 1384–1399, 1992.

[237] J. R. Shewchuk. Delaunay Refinement Algorithms for Triangular Mesh Generation. *Computational Geometry: Theory and Applications*, 22:1–3, 2001.

[238] K. Smith. An argument for abandoning the travelling salesman problem as a neural-network benchmark. *Neural Networks, IEEE Transactions on*, 7(6):1542–1544, Nov 1996.

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

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

[241] SOM. Bibliography of SOM papers. `http://www.cis.hut.fi/research/som-bibl`, 2009.

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

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

[244] S. N. Spitz and A. A. G. Requicha. Multiple-Goals Path Planning for Coordinate Measuring Machines. In *ICRA*, pages 2322–2327, 2000.

[245] M. Strandberg. Augmenting RRT-planners with local trees. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*, volume 4, pages 3258–3262, 2004.

[246] N. Sudha and A. R. Mohan. Design of a hardware accelerator for path planning on the Euclidean distance transform. *J. Syst. Archit.*, 54(1-2):253–264, 2008.

[247] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe. Fast exact and approximate geodesics on meshes. *ACM Trans. Graph.*, 24(3):553–560, 2005.

[248] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM J. Comput.*, 21(5):863–888, 1992.

[249] T. Svoboda. *Central Panoramic Cameras, Design, Geometry, Egomotion*. PhD thesis, Czech Technical University in Prague, Czech Republic, 1999.

[250] A. Tabatabaei and A. Mohades. Clarity Watchman Route. In *The Kyoto International Conference on Computational Geometry and Graph Theory*, Kyoto, Japan, 2007.

[251] X. Tan and T. Hirata. Finding shortest safari routes in simple polygons. *Information Processing Letters*, 87(4):179–186, 2003.

[252] Y.-A. Tan, X.-H. Zhang, L.-N. Xing, X.-L. Zhang, and S.-W. Wang. An Improved Multi-agent Approach for Solving Large Traveling Salesman Problem. In *Agent Computing and Multi-Agent Systems*, pages 351–361, 2006.

[253] J. Tang, G.-S. Wu, F.-Y. Zhang, and M.-M. Zhang. Fast approximate geodesic paths on triangle mesh. *International Journal of Automation and Computing*, 4(1):8–13, January 2007.

[254] S. Thrun. Robotic Mapping: A Survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[255] M. Trevisan, M. A. P. Idiart, E. P. e Silva Jr., and P. M. Engel. Exploratory Navigation Based on Dynamical Boundary Value Problems. *Journal of Intelligent and Robotic Systems*, 45(2):101–114, 2006.

[256] C. Urmson and R. Simmons. Approaches for Heuristically Biasing RRT Growth. In *IEEE/RSJ IROS 2003*, October 2003.

[257] J. Urrutia. *Art Gallery and Illumination Problems*, chapter 22, pages 973–1027. Handbook of Computational Geometry. North-Holland, Amsterdam, Netherlands, 2000.

[258] M. Veeck and W. Burgard. Learning polyline maps from range scan data acquired with mobile robots. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 2, pages 1065–1070, 2004.

[259] R. Vidal, O. Shakernia, H. Kim, D. Shim, and S. Sastry. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. *Robotics and Automation, IEEE Transactions on*, 18(5): 662–669, Oct 2002.

[260] F. C. Vieira, A. ao Duarte Dória Neto, and J. A. F. Costa. An Efficient Approach to the Travelling Salesman Problem Using Self-Organizing Maps. *International Journal of Neural Systems (IJNS)*, 13(2):59–66, 2003.

[261] N. Vishwanathan and I. Wunsch, D.C. ART/SOFM: a hybrid approach to the TSP. In *Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on*, volume 4, pages 2554–2557, 2001.

[262] V. Vonásek. Multiple Traveling Salesmen Problem with MinMax Criterion (Bachelor's thesis), 2006. Czech Technical University in Prague, in Czech.

[263] V. Vonásek. Trajectory generation for cooperative inspection task. Master's thesis, Czech Technical University in Prague, Czech Republic, 2008. in Czech.

[264] V. Vonásek, J. Faigl, T. Krajník, and L. Přeučil. RRT–Path: a guided Rapidly exploring Random Tree. In *Proccedings of ROMOCO'09*, 2009.

[265] J. O. Wallgrün. Hierarchical Voronoi-based Route Graph Representations for Planning, Spatial Reasoning, and Communication. In P. Doherty, editor, *Proceedings of the 4th International Cognitive Robotics Workshop (CogRob-2004)*, pages 64–49, 2004.

[266] P. Wang. *View planning with combined view and travel cost*. PhD thesis, Simon Fraser University, 2007.

[267] R. Wein, J. P. van den Berg, and D. Halperin. The visibility–voronoi complex and its applications. In *SCG '05: Proceedings of the twenty-first annual symposium on Computational geometry*, pages 63–72, New York, NY, USA, 2005. ACM.

[268] R. Wein, J. P. van den Berg, and D. Halperin. Planning High-quality Paths and Corridors Amidst Obstacles. *I. J. Robotic Res.*, 27(11-12):1213–1231, 2008.

[269] K. Williams and J. Burdick. Multi-robot boundary coverage with plan revision. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1716–1723, May 2006.

[270] D. Wolter and K.-F. Richter. Schematized Aspect Maps for Robot Guidance. In *Proceedings of the ECAI Workshop Cognitive Robotics (CogRob)*, 2004.

[271] S. C. Wong and B. A. MacDonald. A topological coverage algorithm for mobile robots. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 1685–1690, Las Vegas, October 2003.

[272] D. T. Wooden. *Graph-based Path Planning for Mobile Robots*. PhD thesis, Georgia Institute of Technology, 2006.

[273] X.-F. Xie and J. Liu. Multiagent Optimization System for Solving the Traveling Salesman Problem (TSP). *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39(2):489–502, 2009.

[274] T. Yamakawa, K. Horio, and M. Hoshino. Self-Organizing Map with Input Data Represented as Graph. In *Neural Information Processing*, pages 907–914. Springer Berlin / Heidelberg, 2006.

[275] M. Yamashita, H. Umemoto, I. Suzuki, and T. Kameda. Searching for mobile intruders in a polygonal region by a group of mobile searchers (extended abstract). In *SCG '97: Proceedings of the thirteenth annual symposium on Computational geometry*, pages 448–450, New York, NY, USA, 1997. ACM.

[276] M. Yamashita, I. Suzuki, and T. Kameda. Searching a polygonal region by a group of stationary k-searchers. *Information Processing Letters*, 92(1):1–8, 2004.

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

[278] A. Yershova. MPNN: Nearest Neighbor Library for Motion Planning. `http://www.cs.duke.edu/~yershova/software/MPNN/MPNN.htm`, 2007.

[279] A. Yershova and S. M. LaValle. Improving Motion-Planning Algorithms by Efficient Nearest-Neighbor Searching. *IEEE Transactions on Robotics*, 23(1):151–157, 2007.

[280] A. Yershova, L. Jaillet, A. Yershova, S. M. LaValle, and T. Siméon. Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain. In *in Proceedings IEEE International Conference on Robotics and Automation*, pages 3867–3872, 2005.

[281] Z. Yu, L. Jinhai, G. Guochang, Z. Rubo, and Y. Haiyan. An implementation of evolutionary computation for path planning of cooperative mobile robots. In *Intelligent Control and Automation, 2002. Proceedings of the 4th World Congress on*, volume 3, pages 1798–1802, 2002.

[282] W. D. Zhang, Y. P. Bai, and H. P. Hu. The incorporation of an efficient initialization method and parameter adaptation using self-organizing maps to solve the tsp. *Applied Mathematics and Computation*, 172(1):603–623, 2006.

[283] A. Zhu and S. Yang. An improved self-organizing map approach to traveling salesman problem. In *Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003 IEEE International Conference on*, volume 1, pages 674–679, Oct. 2003.