Autonomous Robots manuscript No.
(will be inserted by the editor)

Coordinating Heterogeneous Teams of Robots using Temporal Symbolic Planning

Kai M. Wurm · Christian Dornhege · Bernhard Nebel
Wolfram Burgard · Cyrill Stachniss

Received: date / Accepted: date

Abstract The efficient coordination of a team of heterogeneous robots is an important requirement for exploration, rescue, and disaster recovery missions. In this paper, we present a novel approach to target assignment for heterogeneous teams of robots. It goes beyond existing target assignment algorithms in that it explicitly takes symbolic actions into account. Such actions include the deployment and retrieval of other robots or manipulation tasks. Our method integrates a temporal planning approach with a traditional cost-based planner. The proposed approach was implemented and evaluated in two distinct settings. First, we coordinated teams of marsupial robots. Such robots are able to deploy and pickup smaller robots. Second, we simulated a disaster scenario where the task is to clear blockades and reach certain critical locations in the environment. A similar setting was also investigated using a team of real robots. The results show that our approach outperforms ad-hoc extensions of state-of-the-art cost-based coordination methods and that the approach is able to efficiently coordinate teams of heterogeneous robots and to consider symbolic actions.

Keywords Multi-robot coordination · temporal planning

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) in the Transregional Collaborative Research Center *SFB/TR8 Spatial Cognition* projects A3-[MultiBot] and R7-[PlanSpace] and by Microsoft Research, Redmond.

K. Wurm, C. Stachniss, and W. Burgard
University of Freiburg, Dept. of Computer Science, Georges-Köhler-Allee 079, 79110 Freiburg, Germany E-mail: (wurm|stachnis|burgard)@informatik.uni-freiburg.de

C. Dornhege, and B. Nebel
University of Freiburg, Dept. of Computer Science, Georges-Köhler-Allee 052, 79110 Freiburg, Germany E-mail: (dornhege|nebel)@informatik.uni-freiburg.de

1 Introduction

One of the fundamental problems in mobile robotics is the task of autonomously navigating in an environment. In most realistic settings, a robot only has partial knowledge about its environment. For example, the blueprint of a building may be known to the robot but furnishing and other obstacles are usually not known beforehand. Furthermore, there are applications in which the environment is entirely unknown to the robot, for example during planetary exploration and disaster recovery missions.

In many applications, a coordinated team of robots offers advantages over a single robot. Multi-robot systems have the potential of being more fault tolerant and of reducing the overall time to complete a given task [Dudek *et al.*, 1996; Cao *et al.*, 1997]. A well-studied problem is the exploration of an unknown environment with a cooperative team of robots. Previous approaches often addressed the problem of exploring an environment with groups of robots that have identical capabilities. To coordinate such *homogeneous* teams, popular approaches determine a set of exploration targets and assign robots to them numerically [Zlot *et al.*, 2002; Ko *et al.*, 2003; Berhault *et al.*, 2003; Burgard *et al.*, 2005; Stachniss, 2009]. These approaches consider the cost and the expected information gain of each exploration target.

In this paper, we address the problem of coordinating exploring teams of robots with differing capabilities. The robots in such *heterogeneous teams* may differ in their physical properties such as their sensor setup, their size and payload, their maximum traveling speed, or the type of terrain they are able to traverse. In our approach, we especially consider robots that differ in the actions they are able to perform. For instance, robots might be equipped with manipulators, they could be able to deploy localization beacons, or they could even deploy other robots. Numeric, cost-based

coordination approaches are able to consider differing properties of robots that affect navigation costs. For example, the cost of reaching a target depends on the travel speed of a robot and it is also possible to encode that a robot cannot reach a target due to constraints on the size of the robot or the type of terrain it can traverse. However, it is not straightforward to take into account actions other than navigating to exploration targets since there is no efficient mapping of such actions to cost or utility measures. While we can usually specify the time it takes to perform an action such as deploying a robot, it is not meaningful to compare this cost to the cost of exploring a given frontier target. The reason for this incompatibility lies in the fact that performing an action that does not explore parts of the environment has no apparent immediate reward to an exploration system but it effects its performance in the future.

From a conceptual point of view, the ability to perform actions that go beyond navigating to goal positions introduces corresponding *symbolic actions*. This term is commonly used in classical planning approaches that encode the state of a system using logical symbols. Classical planning approaches execute symbolic actions to change the state of the system towards a pre-defined goal state. In the context of multi-robot exploration, we define a symbolic action as any action that does not explore parts of the environment directly. Examples of such actions include opening doors, operating elevators, moving obstacles, and deploying other robots. There exist relatively few approaches that coordinate teams of robots and take into account symbolic actions. One specific set of actions that has previously been considered is the deployment and retrieval of robots by other robots. To execute symbolic actions, previous approaches rely on manually designed strategies [Singh and Fujimura, 1993; Murphy *et al.*, 1999; Dellaert *et al.*, 2002]. One strategy, for example, is to deploy a smaller robot whenever a large robot cannot reach a given goal. These strategies, however, are specific to a certain type of robot and environment and it is unclear whether they are able to efficiently coordinate large teams of robots.

The approach presented in this paper considers symbolic actions explicitly by applying symbolic planning techniques. In the context of multi-robot exploration our goal is to explore all exploration targets as fast as possible. We assume that to reach some targets it is necessary to execute additional symbolic actions such as removing an obstacle or operating an elevator. The key idea of our approach is to integrate a symbolic planning system and a robotic path planner. Since it is unclear how we can translate symbolic actions into a cost measure as it is used in numeric coordination approaches, we instead treat navigating to an exploration target as an action in a symbolic planning system. We generate a symbolic formulation of the coordination problem that includes navigation goals as well as symbolic actions that have to be executed. This description serves as the input to a symbolic planning system that solves the coordination problem. However, classical planning approaches do not consider the execution cost of the solutions they generate. Adding costs measures to actions turns the coordination problem into a *temporal planning problem* and there exist efficient algorithms to solve such problems [Gerevini *et al.*, 2008; Eyerich *et al.*, 2009]. We estimate execution costs for each navigation action using a robotic path planner.

In contrast to cost-based coordination approaches, our system is able to explicitly plan for the execution of symbolic actions. Still, the use of action costs that are determined by the robotic path planner allows us to generate time-efficient solutions. In this way, our approach combines the strength of cost-based coordination approaches with the flexibility of symbolic planning systems. To evaluate our coordination approach, we apply our framework to two popular multi-robot applications: exploration of unknown environments with heterogeneous teams of robots and disaster recovery with heterogeneous teams.

An interesting coordination problem arises when a team of robots includes members that are able to deploy and retrieve other, smaller robots. For a task such as the autonomous exploration of lunar craters, one can imagine robots that approach the crater and then deploy a specialized robot which descents into the crater [Cordes *et al.*, 2011; ESA, 2008]. Such systems are frequently referred to as *marsupial robots* [Murphy *et al.*, 1999]. The coordination of marsupial teams generally requires to carefully plan deployment and retrieval actions, both are symbolic actions according to our definition. In addition, one has to take into account the different properties of the robots such as their sensor setup, their size and payload, their maximum traveling speed, or the type of terrain they are able to traverse. The coordination framework proposed in this paper is applied to the exploration of an unknown environment using marsupial teams of robots.

A further class of applications for teams of heterogeneous robots is centered around the scenario that important parts of an environment have collapsed after a natural disaster. In such a setting, teams of robots can, for instance, be used to perform search and rescue missions. We consider the task of carrying out pre-defined actions at certain critical locations in the collapsed structure, for example, to reestablish safe operation of a failed power station or chemical plant. Such a disaster recovery task requires robots with diverse capabilities. First of all, robots are required to clear collapsed paths throughout the structure. At the same time, there is a need for robots with good sensors or precise manipulation skills to open doors, operate valves, closely inspect parts of the building, or repair damage. A heterogeneous team of specialized robots can be used to provide these capabilities in a flexible way. We apply our proposed

framework to coordinate a team of exploring and clearing robots in a simulated disaster scenario. In addition to exploratory navigation actions, we explicitly plan for symbolic clearing actions.

Both application scenarios serve as motivating examples for this work. We developed and implemented a coordination approach for both applications and compared our approach to ad-hoc extensions of cost-based numeric coordination approaches. As we will show in the evaluation, our approach produces significantly better plans leading to a decrease in both the overall runtime and the sum of traveled distances. Additionally, we applied our approach to coordinate a real-world heterogeneous robotic system.

This paper extends our previous work [Wurm *et al.*, 2010] in several ways. First, it provides an additional application of our approach in the context of disaster recovery. It furthermore contains an extended experimental evaluation. Additionally, we describe an application realized with a real-world heterogeneous robotic system. We finally include a description of temporal planning from a robotics perspective and discuss strengths and limitations of its application in robotics.

The remainder of this paper is organized as follows. After discussing related work, we give a short introduction to temporal planning. In Section 4, we give a general description of our coordination framework and its components. We describe a specific application for exploration with marsupial robots in Section 5. In Section 6, we present a further application of our framework in the context of disaster recovery. Our experimental evaluation is presented in Section 7. Finally, strengths and limitations of the proposed approach are discussed in Section 8.

2 Related Work

When multiple robots explore an unknown environment the coordination strategy has the biggest influence on the performance of the system. Coordination methods for homogeneous teams, in which all members are equipped equally, have been studied extensively in the past. In those cases, the coordination task is often formulated as an assignment problem. Such approaches assign robots to exploration targets based on a suitable cost measure.

Several methods have been presented to determine an assignment of robots to exploration targets. Most methods determine targets by extracting the frontier between explored and unexplored areas. This target generation approach was introduced by Yamauchi [1997] who also proposed a decentralized multi-robot coordination technique that assigns each robot to the nearest frontier [Yamauchi, 1998]. The drawback of this coordination method is that several robots can be assigned to the same target.

To improve the overall performance, later approaches introduced techniques to avoid redundant work. Zlot and colleagues [2002] proposed an architecture in which the exploration is guided by a market economy. They consider sequences of potential target locations for each robot and trade tasks between the robots using single-item first-price sealed-bid auctions. Such auction-based techniques have also been applied by Berhault *et al.* [2003] to assign robots to bundles of targets so that synergy effects between targets are exploited.

Burgard *et al.* [2005] presented an iterative assignment method based on the estimated cost of reaching a target. It additionally considers visibility constraints between targets. Ko *et al.* [2003] and Stachniss [2009] presented algorithms that use the Hungarian method to compute the assignments of robots to exploration targets. In a previous work, we presented approaches that use semantic information [Stachniss *et al.*, 2009] and segmentations of the environment [Wurm *et al.*, 2008] for improving coordinated exploration. By assigning robots to unexplored segments instead of frontier targets, a more balanced distribution of the robots over the environment is achieved and the overall exploration time is reduced.

Heterogeneous teams of robots consist of robots with different abilities. An early approach towards coordination of heterogeneous robot systems during exploration was presented by Singh and Fujimura [1993]. In this approach, if a robot is too big to pass through a narrow passage, other robots are informed about this task. A system for mapping with heterogeneous robots was introduced by Grabowski and Navarro-Serment [2000]. Coordination, however, is performed manually in their approach. Howard *et al.* [2006] presented an approach to coordinate large teams of heterogeneous robots. In their experiments, a small number of sophisticated robots explored an environment and a large number of simple sensor robots was then deployed to serve as a distributed sensor network.

Whenever small robots with low traveling speeds or limited power resources are used in a heterogeneous robot team, it is advantageous to have larger robots transport the smaller ones to avoid a serious penalty in exploration time or power consumption [Murphy *et al.*, 1999]. The larger carrier robots are frequently referred to as *marsupial robots*. One of the first physical implementation of a marsupial system was presented by Murphy *et al.* [1999] who also described ad-hoc methods to deploy the smaller robot. Kadioglu and Papanikolopoulos [2003] presented a further physical implementation of a marsupial system. Rybski *et al.* [2000] developed strategies to use marsupial systems for surveillance. In their approach, small scout robots are deployed by the marsupial robot to monitor individual rooms. Dellaert *et al.* [2002] deployed a team of legged robots using a carrier robot in a rescue scenario. Denner and Papanikolopoulos [2007] presented a deployment method for marsupial

teams that explicitly considers power constraints. In all of the previously described exploration systems, deployment and retrieval of robots in marsupial teams is determined by handcrafted methods. In contrast to that, the approach presented in this chapter explicitly takes these symbolic actions into account when coordinating the exploration.

The approach presented in this paper employs domain independent planning techniques to coordinate multiple robots. Domain independent planning is a sub-field of artificial intelligence that has been investigated thoroughly [Fikes and Nilsson, 1971; Bylander, 1994; Erol *et al.*, 1995] using methodes like action graphs [Gerevini and Serina, 1999], planning as satisfiability [Kautz and Selman, 1992] and heuristic state-space search [Bonet and Geffner, 2001].

A classical planning problem consists of a set of state-variables with finite domains, an initial state, a set of actions and a goal condition. An action is defined by a precondition and its effects, which is a set of variable assignments. A solution for a classical planning problem is then defined as a finite sequence of actions that leads from the initial state to a goal state. There exist several efficient planning systems for classical planning problems [Bonet and Geffner, 2001; Hoffmann and Nebel, 2001; Chen *et al.*, 2006; Helmert, 2006; Richter and Westphal, 2010].

In multi-robot coordination, actions need to be executed concurrently and they usually have variable durations. These temporal constraints cannot be handled by classical planning approaches and constitute a new class of problems. The task of finding solutions to these problems is known as temporal planning and several efficient approaches have been presented [Gerevini *et al.*, 2008; Eyerich *et al.*, 2009]. The predominant approach of solving temporal planning problems is forward search with a search guidance function using A* or similar algorithms. Most approaches support the use of numerical state variables. In contrast to binary and multi-valued state variables, numeric state variables have an infinite continuous value domain. While numeric state variables lead to undecidability even when used in a very limited form [Helmert, 2002], they are considered to be of high importance when modeling real world domains. Temporal planning is more complex than classical planning, which is PSPACE-complete [Bylander, 1994]. Although the general problem of temporal planning is EXPSPACE-complete, wide classes are still in PSPACE [Rintanen, 2007].

While temporal planners generate sequences of abstract actions, most real-world applications in robotics require explicit motion commands that can be executed by the robots. For this reason, a number of approaches have been proposed that integrate other reasoners, such as motion planners, to generate low-level actions. The work by Cambon *et al.* [2009] focuses on the integration of manipulation planning with a symbolic planner. Kaelbling and Lozano-Perez [2011] address the problem of successor generation

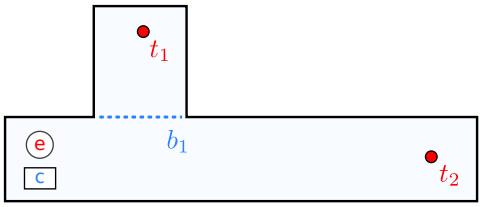

Fig. 1 An exploring robot (e) has to `explore` both targets t_1 and t_2. The path to t_1 is blocked at b_1 (dashed line). A clearing robot (c) can `clear` the blockade.

during planning. They use so called suggesters that provide, for example, new motion paths or goal locations. The work described in this chapter uses the planner TFD/M [Dornhege *et al.*, 2009], a variant of the temporal fast downward planning system originally developed by Eyerich *et al.* [2009]. TFD/M provides a generic interface for the integration of external modules that compute the values of state variables, action costs, and numerical effects during the planning process using sub-processes. By means of these sub-processes, we combine temporal planning with path planners traditionally used for robot navigation and coordination. The work by Gregory *et al.* [2012] uses modules during planning to evaluate not only predicates, but also terms in first order logic.

Autonomous disaster recovery in partially known environments, as introduced in this chapter, has not been studied in depth. An auction-based coordination method as well as a method based on genetic algorithms is presented by Jones *et al.* [2011]. In their work, they coordinate a simulated team of fire trucks and bulldozers leading to a coordination problem that is similar to the disaster recovery domain considered in this chapter. In contrast to our approach, however, blockades are assumed to be known beforehand. It is unclear, how this approach can be extended to partially known environments. Koes *et al.* [2005] described an approach that employs mixed integer linear programming (MILP) to coordinate a heterogeneous team in a search and rescue scenario. In this domain, disaster victims need to be found and identified in a known environment. This task involves assigning robots with different capabilities to appropriate tasks. Similar to our approach, path costs between target locations are explicitly taken into account using a robotic path planner. The static nature of the environment, however, allows all paths to be pre-computed for the entire runtime. Such a pre-computation is not possible in the case of partially known environments.

3 Temporal Planning

Our approach employs a symbolic planning system to compute actions for a team of robots. Algorithms for classical domain independent planning generate plans that consist of sequences of actions. There is a substantial body of literature

on this topic and a description of planning algorithms can, for example, be found in [Russell and Norvig, 2010]. In contrast to these classical planners, temporal planning systems explicitly allow for concurrent actions. Consider the initial state of the simple planning problem shown in Fig. 1. The environment contains two targets t_1 and t_2 and two robots e and c. The task is to explore both targets using robot e. Robot c can clear the blockade b_1 using the action `(clear c b1)` and robot e can explore targets using, for example, the action `(explore e t2)`. Target t_1 is blocked by b_1 and cannot be explored initially.

A classical planner is able to produce a sequence of actions that solves the problem of exploring all targets. For example the following plan is a valid solution:

```
(clear c b1)
(explore e t2)
(explore e t1)
```

This, however, is not the best solution to the task. The first two actions are not dependent on each other, so they can be executed in parallel. Temporal planning allows to express such concurrent actions and would result in the following exemplary plan

```
0.0: (clear c b1)    [1.2]
0.0: (explore e t2)  [2.0]
2.0: (explore e t1)  [1.8],
```

where the actions are preceded by the start time and the durations of actions are given in square brackets. For example, `(explore e t2)` starts at time 0.0 and takes 2.0 time units to complete.

In our approach, we define temporal planning problems using PDDL 2.1 [Fox and Long, 2003], which is a language for defining planning problems and allows for durative actions. Durative actions are an extension of classical planning actions. In addition to conditions and effects they allow the problem description to specify an execution time. Note that solutions to temporal planning problems do not define a strict sequence of actions but merely a partially ordered set and especially that actions might be executed in parallel.

In realistic planning problems, there will be restrictions on which actions can be executed in parallel and which actions have to be completed before another action can be started. These restrictions are modeled as conditions. More precisely, a condition is defined as a tuple $(C_{\vdash}, C_{\leftrightarrow}, C_{\dashv})$, where $C_{\vdash}$ is a start condition that must hold at the start time of the action, $C_{\dashv}$ is an end condition that must hold at the end of the action and $C_{\leftrightarrow}$ is an overall condition that must hold during the execution of the action.

The effects of durative actions are specified in a similar way. An effect is defined as a tuple $(E_{\vdash}, E_{\dashv})$, where $E_{\vdash}$ is a start effect that is applied at the start of the action and $E_{\dashv}$ is an end effect that is applied at the end of the action. For more details on the definition of temporal planning tasks we refer to the work of Eyerich *et al.* [2009].

When coordinating a cooperative team of robots, specific tasks are assigned to individual robots. In most domains, robots work in parallel. Still, their actions might depend on each other, for example, when a robot is clearing a path for another robot. Coordination problems that include such cases can only be solved efficiently when the concurrency of the domain, but also interdependencies of robot tasks, are accounted for. Temporal planning allows for the specification of such conditions and is therefore well suited for multi-robot coordination.

3.1 Planning Domain Definition Language

A wide range of problem types can be modeled as a general planning problem, ranging from transportation problems and single-player games to general combinatorial problems. In recent years, the Planning Problem Definition Language (PDDL) [Fox and Long, 2003] has been established as the prevalent planning language.

To generate a PDDL task description, one needs to define (i) the objects involved in the planning process, (ii) the predicates that define the state of the planner, (iii) actions that change the state, and (iv) a start state and a goal condition. We will give several examples of PDDL statements in the following sections. The PDDL description then forms the input to symbolic planners.

In our approach, PDDL is used to encode the coordination problem that has to be solved in a multi-robot system. Based on this description, the temporal planner computes concurrent actions for the team of robots. We use PDDL/M [Dornhege *et al.*, 2009], an extension to PDDL that allows for the definition of external module calls. Using this method, we combine symbolic planning and a robotic path planner in our approach (see Section 3.3).

3.2 The TFD/M Planning System

TFD/M is a domain-independent progression search planner developed by Dornhege *et al.* [2009]. It extends the temporal planner framework Temporal Fast Downward (TFD) by Eyerich *et al.* [2009]. TFD in turn is based on a classical planning system called Fast Downward that was developed by Helmert [2006]. TFD extends the original system to support durative actions and numeric expressions while TFD/M adds support for external modules.

TFD/M solves a planning problem in three phases: First, the PDDL planning task is translated from its original encoding into a more concise representation using finite-domain variables. This is used by the planner to guide the search by employing hierarchical dependencies between

state variables and leads to an increased search performance. In the second step, efficient internal data structures are generated that are used by the search component and the search guidance function. The most important ones are *domain transition graphs* for each variable that encode how state variables can change their values and the *causal graph* that represents the hierarchical dependencies between different state variables. Finally, a best-first progression search is performed, guided by a numeric temporal variant of the context-enhanced additive method [Helmert and Geffner, 2008].

In contrast to several other temporal planning systems, TFD/M does not decompose the search into an action selection phase and a scheduling phase but searches directly in the space of time-stamped states. This usually leads to plans of significantly higher quality [Eyerich *et al.*, 2009]. Note, however, that the first plan that is generated by this method is not necessarily optimal. This is because the search guidance function is inadmissible, in other words, it does not guarantee an underestimation of the true execution cost of a plan.

TFD/M is implemented as an anytime algorithm that does not terminate after the first solution is generated. It produces a potentially non-optimal solution quickly and then prunes the search space to those time-stamped states which can potentially be extended to solutions with a lower overall duration. If all states in the resulting state space are expanded, the produced solution is guaranteed to be optimal.

3.3 Semantic Attachments in TFD/M

TFD/M features semantic attachments that are a means of evaluating components of the planning task externally. This is implemented as a module interface for predicates, numerical effects, and durations. In our coordination algorithm, semantic attachments are used to specify durations of actions in the planning task description. Consider, for example, the following module specification:

```
(costClear ?r - robot ?b - blockade cost
  costClear@libcost_module.so)
```

This defines a semantic attachment for cost computation (indicated by the `cost` keyword) and is used to specify the duration of actions. It has two parameters, a robot *r* and a blockade *b*. To define a semantic attachment, a function call is provided that performs the actual computation externally (here, `costClear@libcost_module.so`).

We can now use the semantic attachment defined above to specify the duration of actions. The `clear` action in the initial problem (see Fig. 1), for example, can be defined in the following way:

```
(:durative-action clear
 :parameters (?c - clearer ?t - blockade)
 :duration (= ?duration [costClear ?c ?t])...
```

Note that the use of a semantic attachment is indicated by squared brackets in the definition. For further details on the implementation of semantic attachments in forward chaining state space planners such as TFD/M, we refer the reader to our previous work [Dornhege *et al.*, 2009].

When the planner expands actions in the planning phase, it detects semantic attachments and executes the associated dynamic library calls. These external calls compute the appropriate action costs. They are made exaclty at the same time that a planner without modules would acquire the value from the internal state. In our approach, for example, the cost of navigation actions are computed by a robot path planner.

In the general case, those module calls can influence the complexity of the system. Wide classes of temporal planning problems lie in PSPACE, while temporal planning is EXPSPACE-complete [Rintanen, 2007]. The scenarios in this paper belong to those classes, as they do not allow the same action to overlap with itself. This means, that as long as the sub-problem solved by a single module call is in PSPACE, the complete system will retain its complexity. This is due to the fact that module calls depend only on the planner state, but not on other module calls. Thus, once a module call is finished, its memory can be freed and therefore no space is accumulated in memory. The numeric planner used in this work fulfills this condition as it only accumulates a polynomial amount of states – the grid cells.

4 Coordination of Heterogenous Teams of Robots Using Temporal Planning

In the following, we will describe our coordination framework for heterogeneous teams of robots. We assume that to solve a given task, the robots have to move in the environment and additionally need to perform symbolic actions. In this context, symbolic actions are defined as actions that change the state of the overall system but whose primary purpose is not to change the position of the robots. Numeric coordination approaches usually consist of a method to evaluate the utility of navigation goals and a target assignment technique. Since it is unclear how we can translate symbolic actions into a utility measure as it is used in numeric coordination approaches, we instead treat navigating to an exploration target as an action in a symbolic planning system. In our approach, the target evaluation and assignment modules of numeric approaches are replaced by a temporal symbolic planner, a method to generate a problem description for this

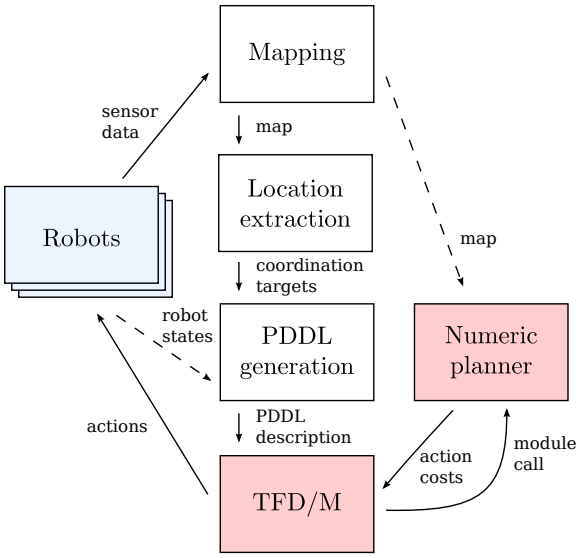

Fig. 2 This figure shows an overview of the coordination system. Solid arrows mark the control flow and dashed arrows represent data passed between modules.

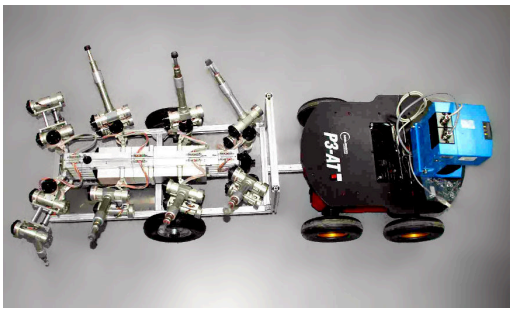

Fig. 3 Example of a marsupial robot team. A versatile but slow legged platform is deployed by a faster wheeled robot. Courtesy of DFKI Robotics Innovation Center.

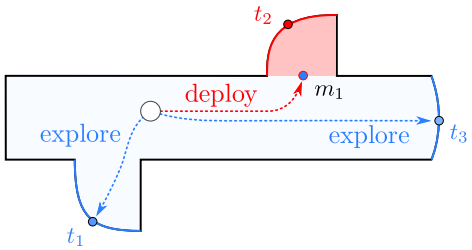

Fig. 4 An exploring robot (white circle) has to choose between three possible actions: explore target t_1, explore target t_3, or deploy a smaller robot at m_1 to let it explore t_2 in the red area that it cannot explore itself.

planner and a numeric path planner that is used to estimate the path costs of navigation actions.

The architecture of our system is illustrated in Fig. 2. In our approach, we assume global and unlimited communication between the robots and thus employ a centralized coordination approach. Furthermore, all robots are assumed to know their individual position. The team of robots provides the sensor data and states of the platforms, such as their positions, current actions and navigation goals, to our centralized coordination system. We use the sensor measurements and poses of the robots to build a grid map of the environment. From this map, we extract locations that are relevant for coordination. In an exploration task, relevant locations would be exploration targets that can, for example, be determined using the frontiers approach [Yamauchi, 1997]. To execute symbolic actions we furthermore extract positions at which the actions need to be executed. If some of the robots were able to open doors, for example, we would determine the positions of doors in the map.

We solve the coordination problem of assigning actions to the robots using the temporal symbolic planning system TFD/M. From the current state of the robots and the locations we extracted from the map of the environment we generate a problem description in PDDL. This description serves as the input for the planner. Since the goal is to solve the given task as fast as possible, we need to consider the costs of traveling to navigation goals. To this end, we make use of the modular interface of TFD/M to call a numeric path planner for mobile robots. The numeric planner returns the estimated path cost of reaching a goal position from a given start position. Based on these estimates, the temporal planner is able to compute an efficient plan that solves the mission goal. From the solution of the symbolic planner, we extract actions and send them to the robots. The loop depicted in Fig. 2 is executed constantly: Whenever new information about the environment arrives, for example, new sensor data is perceived or one of the robots reaches a target, we replan.

Some of the modules in our coordination framework are specific to the application. Depending on which actions the robots are able to perform, we need to adapt the PDDL description of the coordination problem. The possible actions furthermore influence which locations we need to extract from the map. In the following, we will present two applications and describe how to adapt the framework to them.

5 Coordination of Marsupial Teams

In the first application, we use our planning framework to coordinate a team of marsupial robots. In such a team, one group of robots, the *carriers*, are able to carry, deploy and retrieve a group of other robots, the *rovers*. An example of a real carrier and rover robot is depicted in Fig. 3. Here, a versatile but slow legged platform is deployed by a faster wheeled robot.

We assume that carriers and rovers have different navigation capabilities and that certain areas of the environment can only be explored by the rovers and others only by the carriers. We furthermore assume that the robots are able to

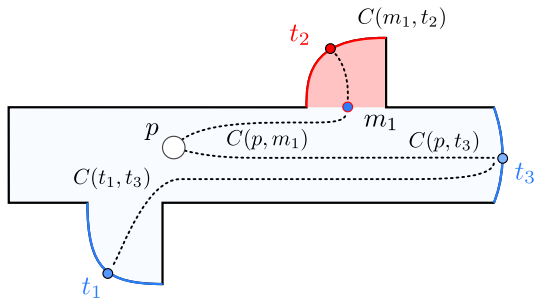

Fig. 5 Example of the costs that have to be considered. Dotted lines illustrate the estimated path costs between the robot at position p and the different target positions t_i, the costs between meeting points m_i and robot or targets positions, and costs between target positions. For the sake of better visibility, we did not display all costs in this figure.

determine which areas are traversable by which robot based on their sensor observations, for example based on techniques developed in our previous work [Wurm *et al.*, 2009].

In our experiments, a marsupial team consists of n carrier robots, where each carrier initially carries m rover robots. The goal is to completely explore the environment, that is, to cover the traversable area with the sensors of the robots. Fig. 4 depicts a situation where a carrier has to choose between exploring the environment itself and deploying a rover in an area that it cannot reach. This illustrates the key challenges that the coordination method faces: It needs to generate exploration targets, to assign robots to those targets, and to schedule deployment and retrieval actions.

5.1 Target Locations and Cost Estimation

To identify potential target locations, a set of exploration targets T is generated from the partially explored grid map. In addition to this, a set of meeting points M is determined. These meeting points are situated at the border between those parts of the environment that can only be traversed by the rovers and the parts that can only be traversed by the carriers. They are used for deployment and retrieval of the rovers (see Fig. 5 for an illustration). To determine the targets and meeting points a frontier extraction algorithm is used as first described by Yamauchi [Yamauchi, 1997].

There are two basic types of actions a carrier can perform: exploring a target or visiting a meeting point to deploy or retrieve a rover (see Fig. 4). While deployment and retrieval are assumed to have constant costs, the cost of traveling between two locations in the environment is defined as the estimated travel time of a given robot. This cost depends on the path length as well as on the traversability constraints and travel speed of the corresponding robot. Let *type* be a robot type (here: carrier or rover), x a location in the environment and $t \in \{T \cup M\}$ a target. We define the cost for reaching t as:

$$C_{type}(x,t) \quad (1)$$
$$= \begin{cases} \text{est. path cost}(x,t) & \text{, if robots of type } type \\ & \text{can reach } t \text{ from } x \\ \infty & \text{, otherwise.} \end{cases}$$

Finally, the exploration task is assumed to be completed as soon as the set of exploration targets T is empty.

5.2 Formulating the Exploration Problem as a Temporal Planning Problem

To apply the coordination approach described above, we need to encode the coordination problem that arises in a given situation as a PDDL description. First, we define which types of objects are involved. In the exploration scenario, possible objects are robots that can be either rovers or carriers and locations that can be meeting points or exploration targets. The corresponding PDDL statements are given in Fig. 6 (left). Second, we specify the predicates that are used to define the state. The major predicates we use to describe the exploration problem are

`(at ?r - robot ?x - location),`

which describes that the robot r is at position x and

`(on ?e - rover ?c - carrier)`

which is used to determine whether a rover e is docked at a carrier c. For each target $t \in T$, we also define if it has been explored

`(explored ?t - target).`

Additionally, we use a numeric state variable

`(num_docked ?c - carrier)`

that keeps track of the number of rovers that are docked at a carrier c.

Third, the actions that change the state are provided. We define four actions, namely `dock`, `undock`, `move`, and `explore`. The actions `dock` and `undock` are used to deploy or pick up a rover. They require that the carrier and the rover are at the same meeting point, which is ensured using the `at` predicate. For docking, the number of docked rovers has to be lower than the carrier's capacity and the action changes a rover's state from being at a meeting point to being on a carrier (encoded using the `on` predicate).

The other two actions `move` and `explore` model the possible motions of the robots. The `move` action moves a robot to a meeting point for deployment or retrieval while the `explore` action moves the robot to a target and explores it. To define the duration of the `move` and `explore` actions, we employ the module interface of the temporal planner. Instead of specifying a constant duration or a fixed formula, we call an external module that determines the duration of the action. In our setting, the external module is

```
(:types
  robot
  carrier rover - robot
  location
  target meeting - location )
(:predicates
  (at ?r - robot ?x - location)
  (on ?e - rover ?c - carrier)
  (explored ?t - target)
  (can_explore ?r - robot ?t - target))
```

```
(:durative-action explore
  :parameters (?r - robot
     ?s - location ?g - target)
  :duration (= ?duration
     [pathCost ?r ?s ?g])
  :condition (and (at start (at ?r ?s))
     (at start (not (explored ?g)))
     (at start (can_explore ?r ?g)) ... )
  :effect
    (and
      (at start (not (at ?r ?s)))
      (at end (at ?r ?g))
      (at start (explored ?g))
      ... ))
```

```
(:init
  (at robot0 p)
  (on robot1 robot0)
  (can_explore robot0 t1)
  (can_explore robot1 t2)
  (can_explore robot0 t3)
)
(:goal (and
  (explored t1)
  (explored t2)
  (explored t3)
))
```

Fig. 6 Examples of PDDL definitions. Left: definition of the types and predicates used in the coordination domain. Middle: definition of the `explore` action. Right: example that shows how to specify the current state of the world for the TFD/M planner (see illustration shown in Fig. 4).

realized by an efficient path planner for mobile robots that plans the optimal trajectory of the robot to the given target location based on the current occupancy grid map constructed by the robots so far. Fig. 6 (middle) depicts the PDDL statements that describe the action `explore`. The term `[pathCost ?r ?s ?g]` represents the call to the external module. In our current implementation, we use Dijkstra's algorithm for cost estimation as it allows to efficiently compute the path cost from a given start position to all goal positions. Finally, the initial state of the current planning procedure and the goal state are specified. For the situation depicted in Fig. 4, this is exemplified in Fig. 6 (right).

6 Coordinated Disaster Recovery

As a further application of the coordination approach described in Section 4, we investigate teams of robots that operate in a disaster recovery scenario. More specifically, we assume that a known building is partially collapsed so that paths within the building are blocked at unknown positions. The goal is to visit a given set of targets in the building using a team of robots. In a real world application, the robots could, for instance, provide high-resolution views of the situation, close a set of valves, or deploy a set of beacons. We assume that there are two types of robots: *exploring robots* that visit targets by performing a specific action at a given location and *clearing robots* that are able to clear blocked paths. Both types of robots are able to detect blockades using their sensors. All robots are provided with a map of the environment that includes the mission targets. This prior map does not, however, include blocked paths.

The key challenge in this scenario is to coordinate the team of robots so that the exploring robots do not waste time by waiting for blocked paths to be cleared. An illustration of a typical situation in this problem domain is given in Fig. 1.

6.1 Target Locations and Cost Estimation

The initial mission map includes all target locations that have to be visited. These locations are extracted by the coordination module and stored in a set $T = \{t_1, \ldots, t_n\}$. Whenever a target is visited by one of the exploring robots, it is removed from the set T and the mission map.

While the robots navigate in the environment they update their maps using their sensors. Whenever a blockade b is sensed, this information is distributed to all other robots. The central coordinator keeps track of blockades in a set $B = \{b_1, \ldots, b_m\}$ which is used to coordinate the clearing robots.

Furthermore, the coordination system maintains a representation of the environment that is composed of the prior building map and the set of blockades sensed by the robots. This map is used by the robot path planner to plan collision free paths for the team of robots and to estimate travel costs during target assignment. It estimates path costs from a location x to a target $t \in \{T \cup B\}$ according to

$$C(x,t) = \begin{cases} \text{est. path cost}(x,t) & \text{, path from } x \text{ to } t \text{ traversable} \\ \infty & \text{, path blocked.} \end{cases} \quad (2)$$

This is an optimistic estimate, as we do not assume any knowledge about where blockades may appear. When a probabilistic model of blockades is given the path planning problem can be formulated as the Canadian Traveller's Problem [Papadimitriou and Yannakakis, 1991]. There exist efficient methods to solve this problem [Eyerich *et al.*, 2010] that can be used for the path cost estimation. Note, however, that in our formulation blockades can be cleared and thus this estimate would also not be optimal.

A disaster recovery mission is considered completed as soon as the set of targets T is empty, that is, all of the targets have been visited by an exploring robot.

6.2 Formulation as a Temporal Planning Problem

To apply our coordination approach to the disaster recovery problem described above, we encode the system's state and possible actions using PDDL statements. This PDDL description then serves as input to our coordination algorithm as depicted in the system overview in Fig. 2.

```
(:types
  robot - object
  clearer explorer - robot
  location - object
  free_location - location
  mission_target - free_location
  blockade - location)
(:predicates
    (at ?r - robot ?x - location)
    (cleared ?t - blockade)
    (explored ?t - mission_target))
```

```
(:durative-action explore-from-blockade
  :parameters (?r - explorer ?s - blockade ?g - mission_target)
  :duration (= ?duration [pathCost ?r ?s ?g])
  :condition (and (at start (at ?r ?s)) (at start (not (explored ?g)))
     (at start (not (= ?s ?g))) (at start (cleared ?s)) )
  :effect
    (and
      (at start (not (at ?r ?s)))
      (at end (at ?r ?g))
      (at start (explored ?g)) ))
```

Fig. 7 Examples of PDDL definitions used in the disaster recovery domain. Left: definition of the types and predicates. Right: definition of the `explore-from-blockade` action.

First, we define which types of objects are involved. In this scenario, the relevant objects are the robots that can be either exploring robots or clearing robots and the map locations that can be mission targets or blockades. Fig. 7 (left) shows the corresponding PDDL statements. Second, we specify the predicates that we use to define the problem states. The state in the disaster recovery problem can be specified using

`(at ?r - robot ?x - location)`

which describes that the robot r is at position x,

`(cleared ?b - blockade)`

which describes that blockade b has been cleared, and

`(explored ?t - mission_target)`

which describes that mission target t has been visited by an exploring robot.

Third, we provide the actions that change the state of the system. There are four kinds of movement actions: We distinguish between actions that explore a mission target and actions that have a goal position other than a mission target. We furthermore differentiate actions whose start location is a blockade and actions that start at a location in free space. As an example, the PDDL definition given in Fig. 7 (right) defines a movement action that explores a target starting from a blockade. It can be seen, that movement actions which start at a blockade require the blockade to be cleared before execution. All movement actions require the robot to be at the start location and result in the robot being at the goal location. The costs of these actions are defined as the estimated path costs and are determined via the modular interface much the same as in the marsupial exploration scenario. The goal location of all exploration actions is a mission target t. Once such an action is executed, the current state is changed so that the corresponding predicate `(explored t)` is set to true.

Movement actions that start at a blockade have the additional precondition that the blockade is `cleared`.

In addition to exploration actions, we define a `clear` action to coordinate the group of clearing robots. This action expresses that a given blockade b is cleared by a clearing robot that is positioned at the corresponding location. The duration of the action is defined by the time the robot takes to clear the blockade. In our experiments we assume that this value depends on the size of the blockade. After the action is executed, the predicate `(cleared b)` is set to true in the current state. Finally, the initial and goal state are defined. In the initial state, no blockades have been discovered and none of the mission targets have been explored. The goal state is defined as the state that describes all mission targets as explored.

The coordination loop depicted in the system overview (see Fig. 2) is executed continuously until the goal state is reached. While the description of the actions remain unchanged during consecutive planning cycles, the current state of the system is updated to reflect the current state of the system and the environment. The updated state description defines all robots to be at their specific locations and blockades are added to the state whenever they are discovered. Previously cleared blockades or targets that have been explored are irrelevant to the current problem and thus are ignored in the problem description.

7 Evaluation

The approach described in this paper has been implemented and evaluated thoroughly using a multi-robot simulation system. The experiments are designed to show that explicitly planning symbolic actions leads to a significantly more efficient coordination than using a heuristic extension of previous coordination approaches. We evaluated our coordination framework in the two problem domains introduced in Section 5 and Section 6. In addition, we show that our approach is able to produce efficient coordination plans in reasonable time and thus can be employed in a real-world system.

7.1 Simulation System

To evaluate our coordination approach quantitatively, we developed a simulation system that is able to simulate large teams of heterogeneous robots. In our current system, we also simulate laser range sensors. Sensor and odometry noise are not considered since we focus on the coordination aspects of the problems. The environment is modeled using

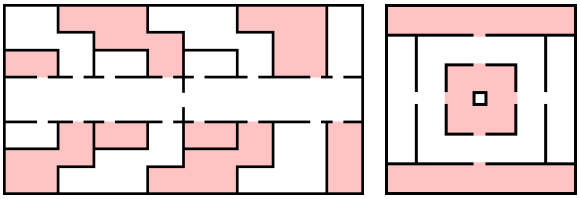

Fig. 8 Simulated environments in the marsupial exploration experiments: *office* (left) and *maze* (right). White areas can only be traversed by carriers while red (dark) areas can only be explored by rovers.

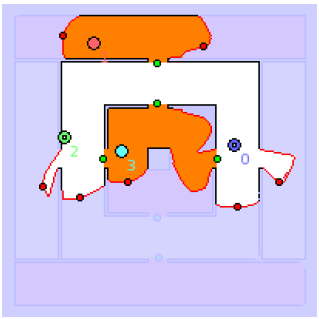

Fig. 9 Visualization of a simulated run in the *maze* environment. Robots are depicted as disks with id numbers and carriers are marked by a white dot. Small circles depict exploration targets (red) and meeting points (green). In this situation, rover 3 could be retrieved by carrier 0 or carrier 2 at any of the three meeting points in its vicinity.

a grid map with additional traversability information and semantic annotations such as mission targets or blockades.

7.2 Exploration with Masupial Teams

We simulated teams of marsupial robots to evaluate our approach introduced in Section 5. The simulated environments contain areas that can only be traversed by rovers and areas that can only be traversed by carriers. The rover robots in most real marsupial systems are considerably slower than the carrier robots. In the evaluation we account for this difference by simulating carrier robots that are twice as fast as rovers. Furthermore, the maximum sensor range of the carriers is also twice as far as the sensor range of the rover robots.

We evaluated robot teams of varying sizes and different environments have been used in the simulation. Two of the environments we used in our experiments can be seen in Fig. 8. The *office* environment resembles a typical office building with two corridors and a number of rooms. Some of the rooms can only be explored by rovers, those are depicted as red areas in the maps. The second environment, in the following referred to as the *maze* environment, features a central area that can only be explored by rovers. In contrast to the areas in the *office* environment, it has multiple meeting points that can be used for deployment. A visualization generated by the simulator is depicted in Fig. 9. It shows a representative coordination problem where exploration targets as well as symbolic actions need to be considered.

To get an intuition of the extend of the environment, one can look at the ratio between the size of the environment and the maximum sensor range of the robots. In many real office buildings, such as building 079 on the Freiburg campus, this ratio is around 10 for a maximum sensor range of 4 m. The ratio for the *maze* environment is 10.2 for rovers and 5.1 for carriers. In the considerably bigger *office* environment the ratio increases to 23.8 and 11.9 respectively.

In both environments, we simulated 30 exploration runs with random initial robot positions. Exploration targets were determined using the frontiers approach and neighboring exploration targets were clustered using visibility constraints similar to the approach proposed by Burgard *et al.* [2005].

7.3 Baseline Approach

We compared our coordination algorithm to a heuristic extension of a method that assigns robots to target locations based on cost estimates [Stachniss, 2009]. In this approach, the Hungarian method is used to compute the cost-optimal assignment of robots to exploration targets. To adapt this method to the problem of exploration with marsupial teams, we first assign the carriers to exploration targets independent of whether they can access those targets or not. The assignment is solely based on the estimated travel cost of the carriers, ignoring traversability constraints in the estimation. The rovers are then deployed as follows: Whenever a carrier is assigned to a target that it cannot explore itself, it will move to the nearest connecting meeting point and deploy a rover there. This rover then explores the targets reachable from the meeting point. As soon as it has finished exploring them, it returns to the meeting point. In our experiments, we assume a limited number of rovers per carrier. This will lead to situations in which a carrier needs to deploy a rover but has none available. The baseline approach then requires the carrier to first retrieve a rover.

The baseline approach as described above closely resembles heuristic deployment rules that have been applied in previous marsupial systems [Murphy *et al.*, 1999]. Note that the baseline approach could be improved by introducing more complex reasoning. For example, it could anticipate cases in which no rovers are available to a carrier and then avoid assigning this carrier to targets that are only reachable by rovers. This would introduce additional state variables that the coordination system would need to maintain. In fact, this kind of reasoning would approximate a planning method such as the one we apply in our approach.

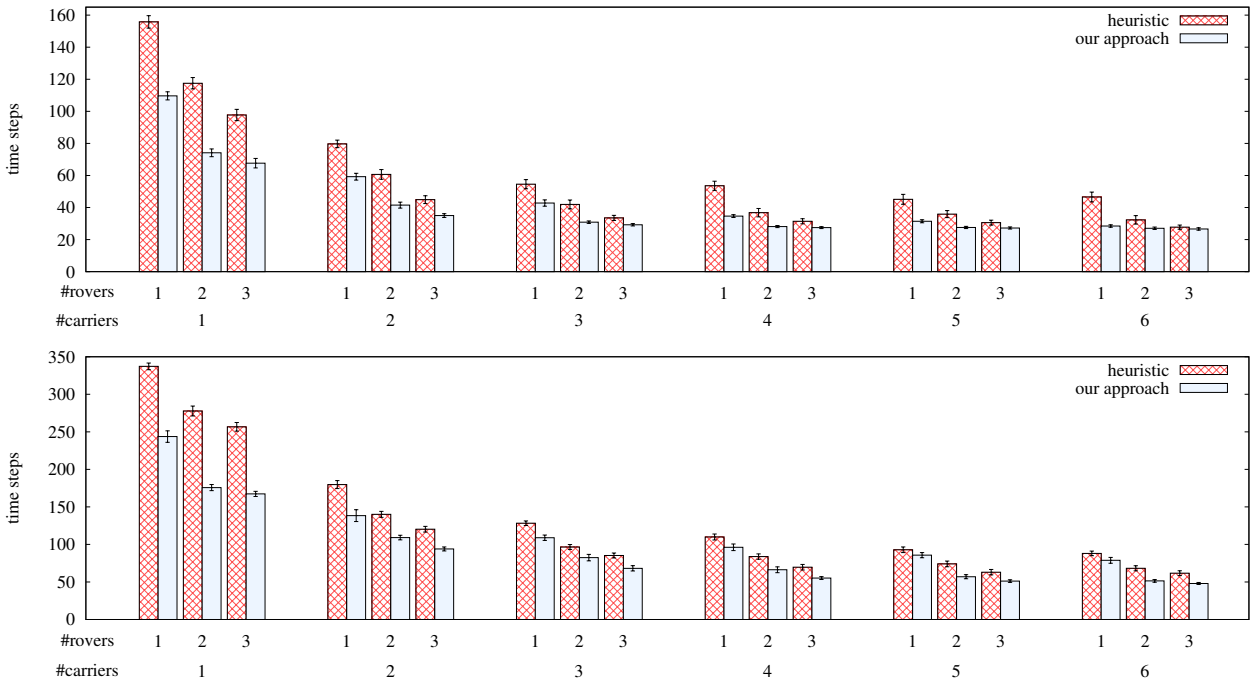

Fig. 10 Exploration time obtained with our approach compared to the ad-hoc method in the *maze* environment (top) and in the *office* environment (bottom) for varying team sizes (number of carriers and number of rovers per carrier). The error bars indicate the 95% confidence intervals.

7.4 Results

An overview of the results obtained in the experiments is given in Fig. 10. It can be seen that our approach explores the environment significantly faster than the baseline method in all configurations. By considering deployment and pick-up actions explicitly, our approach avoids long travel paths and detours that result from the pick-up heuristic in the baseline. In addition, it uses its larger planning horizon to plan sequences of actions while the baseline approach computes exactly one action per robot. Even though we execute only the first action in the sequence, robots often are in a good initial position for the next action.

Especially smaller teams of up to six robots profit considerably from our coordination method. Larger teams can, to some degree, compensate inefficiencies of the coordination when a good distribution in the environment is achieved and many targets can be approached simultaneously. In the settings we evaluated, this effect can be noticed when more than three carriers are used. The number of robots for which this effect occurs clearly depends on the structure and size of the environment.

To evaluate the amount of unnecessary movement in larger teams, we evaluated a further benchmark. We computed the exploration quality according to [Zlot *et al.*, 2002] as

$$\mathscr{Q} = \frac{1}{A} \sum_{i=1}^{n} d_i, \tag{3}$$

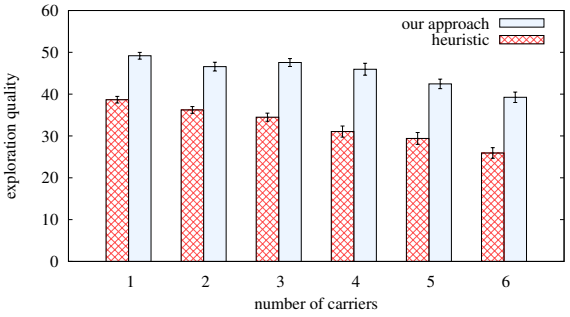

Fig. 11 Exploration quality in the *office* environment over 30 runs using two rovers per carrier. The higher the value, the better the coordination. The error bars indicate the 95% confidence intervals. Note that similar results were obtained for the *maze* environment.

where A is the total area of the environment and d_i denotes the distance traveled by robot i. This measure can intuitively be understood as the average area each robot explores per movement.

The results given in Fig. 11 show that our approach reaches a significantly higher exploration quality. Especially larger teams of robots are coordinated more efficiently, so that unnecessary movements are avoided. The reason for this improvement lies in the larger planning horizon of our approach compared to the baseline. Our approach anticipates future states of the system, especially the position of the robots, while the baseline approach computes actions solely on the basis of the current state of the system. Avoiding unnecessary movements is a significant advantage on its own

since there usually is a direct correlation between the distance traveled and the power consumption of the vehicles. In this way, our approach will save resources in larger teams.

7.5 Coordinated Disaster Recovery

To evaluate the performance of our coordination approach in the disaster recovery domain, we simulated teams of exploring and clearing robots. While the exploring robots are able to visit mission targets, the clearing robots are able to clear blocked paths in the environment.

Several environments have been evaluated and two of those can be seen in Fig. 12. The first environment resembles a hospital with long corridors and a large number of rooms. A set of 15 mission targets and 18 blockades are distributed throughout the building. The ratio between the size of the environment and the maximum sensor range of the robots is 27.05. The second environment is based on the blueprint of a nuclear power plant. Here, 13 targets are grouped around the two reactor cores and control rooms. Paths are blocked at 58 locations throughout the building. The sensor ratio in this fairly large environment is 65.9.

We simulated teams of one to four exploring robots that are accompanied by one to four clearing robots. Simulated clearing robots and exploring robots have the same maximum sensor range and driving speed. The time to clear a blockade depends on its size and can be determined by the robots using their sensors. For each team size, we simulated 30 runs starting at random positions.

7.6 Baseline Approach

We compared our coordination approach to an ad-hoc extension of a cost-based coordination method. In this baseline approach, the exploring robots are assigned to mission targets using the Hungarian Method similar to the approach proposed by Ko *et al.* [2003]. Intuitively, this method assigns each exploring robot to the closest mission target while avoiding to assign multiple robots to the same target. The assignment process is repeated whenever one of the robots reaches a target, a path is sensed blocked, or a blockade has been cleared.

As soon as blockades are discovered in the environment, clearing robots are assigned to clear those blockades. This assignment is determined using the Hungarian Method based on the estimated travel time to the blockades. Note that in the baseline approach, the coordination of the clearing robots does not depend on the mission targets or the assignment of the exploring robots.

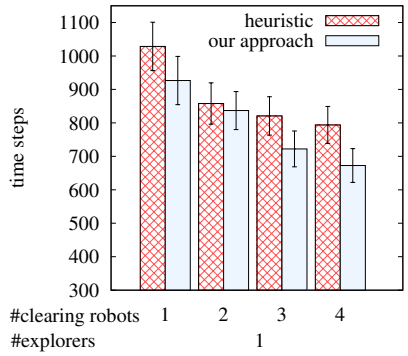

Fig. 14 Time to visit all mission targets using our approach compared to the result using the baseline method in the *power plant* environment with 20% fewer blockades. The error bars indicate the 95% confidence intervals.

7.7 Results

The results obtained in the evaluation of the disaster recovery experiments are shown in Fig. 13. In the *hospital* environment, our approach performed significantly better in all simulated settings. By planning sequences of clearing and exploration actions our approach is able to minimize the time that clearing robots spent waiting for blockades to be cleared.

Similar results could be obtained in the *power plant* map. Due to the large extent of this map and the random placement of start locations, there is a higher variance in the overall runtimes. Using the paired t-test, however, a significant improvement of our approach over the baseline approach could be shown in this map, too.

It is worth mentioning that a significant improvement could not be shown for all simulated team sizes in the *power plant* map when the number of blockades was reduced by 20% and the team size was small (in our experiments: one explorer and two clearing robots). In this special case, the independent assignment of clearing and exploring robots using the Hungarian method provided comparable overall runtimes (see Fig. 14). The low number of blockades enables the baseline approach to simply clear all blockades independent of whether the exploring robot needs a particular path cleared. For this reason, our coordination approach does not profit from its additional lookahead and the ability to plan sequences of actions to the same degree as it does in the more complex settings.

7.8 Planner Runtime

As TFD/M, the temporal planner used in our approach, cannot guarantee optimality, we seek to determine close-to-optimal plans. To analyze the tradeoff between planner runtime and plan quality, we simulated a recovery mission us-

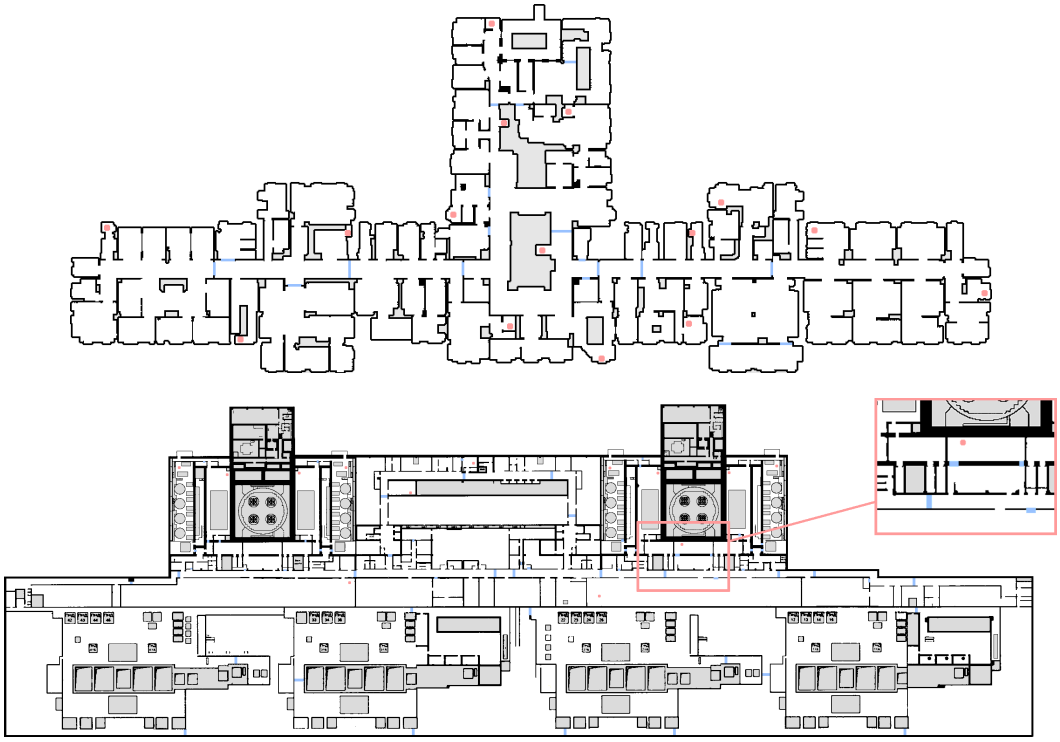

Fig. 12 Simulated environments in the disaster recovery experiments: *hospital* (top) and *power plant* (bottom). Mission targets are visualized by red dots, blockades are shown as blue lines. Gray areas can not be accessed by the robots. In the experiments, blockades were not known to the robots a priori but could be detected by their sensors.

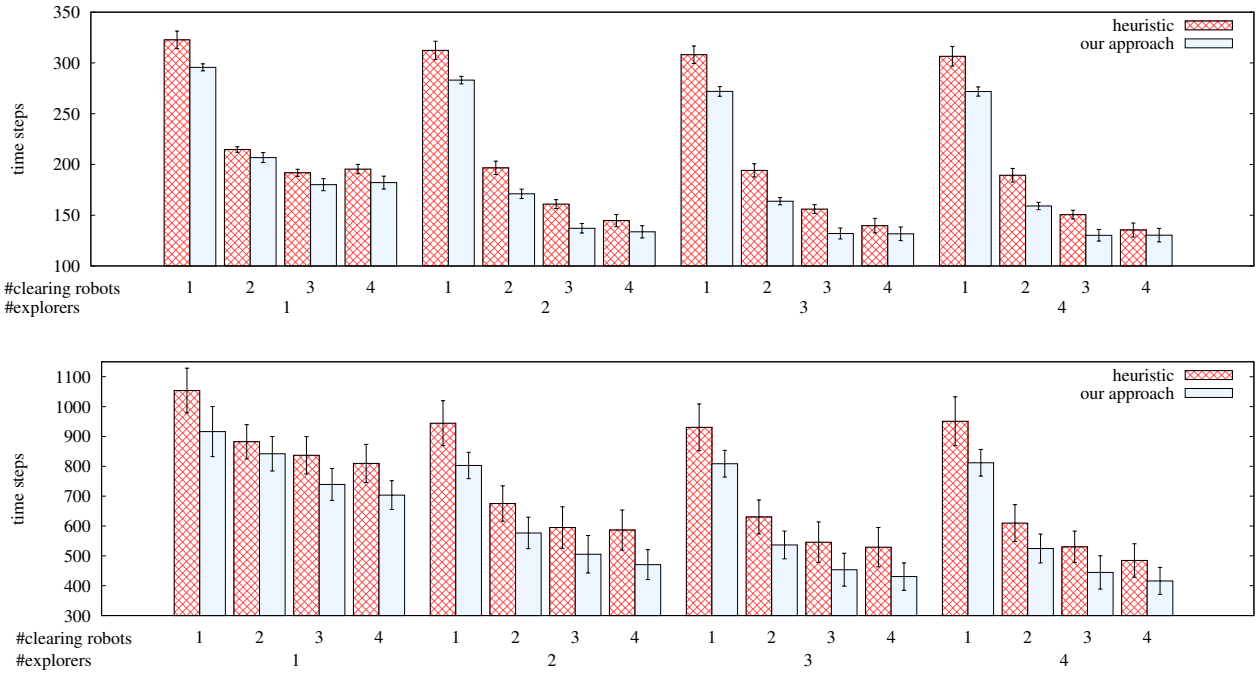

Fig. 13 Runtime until all mission targets are visited using our approach compared to the result using the ad-hoc method in the *hospital* environment (top) and in the *power plant* environment (bottom) for varying team sizes (number of exploring and clearing robots). The error bars indicate the 95% confidence intervals.

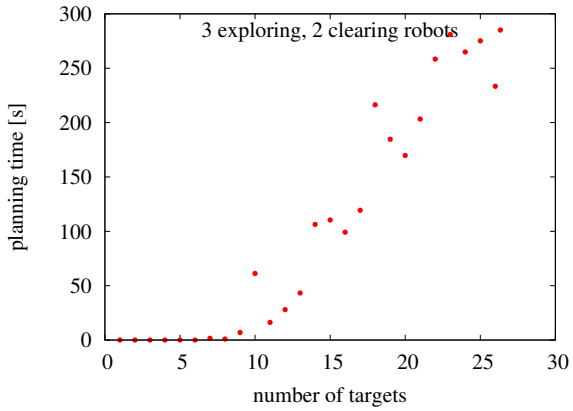

Fig. 15 Average runtime until a plan is computed that is at most 5% worse than the best plan computed until a timeout of 900 s is reached. For this evaluation, three exploring and two clearing robots were coordinated in the *power plant* map.

ing a team of three exploring and two clearing robots in the *power plant* map. In this experiment, the planner was given a maximum planning time of 900 s and we measured the time until a plan was generated that was at most 5% worse than the best plan found until the timeout was reached. The experiment was performed on a 2.7 GHz AMD Opteron 2384 system using one core per task.

The result of this evaluation is shown in Fig. 15. It can be seen that the number of planning targets (exploration targets and blockades combined) has a dominant influence on the planning time. It can also be seen that for a team size of five robots the close-to-optimal plan can be found in less than around 10 s as long as there are less than ten planning targets. Note however that a feasible plan is usually found much faster than the best plan and in all our experiments, it was found within a 30 s planning limit.

7.9 Real World Application

In this experiment, we applied our approach to coordinate a real team of robots. The team consisted of an ActiveMedia Pioneer 2AT equipped with a tilting laser scanner (the explorer) and of a Telemax mobile manipulation platform (the manipulation robot). The goal of the robots was to explore an area which was reachable through a narrow passage that only the exploring robot could traverse. In our experiments, the passage could be free or it could be blocked by movable obstacles. The explorer was able to detect such obstacles and to inform the coordination module about the blockade. The manipulation robot was able to grasp detected obstacles and to remove them from the passage. Once the goal area was reached by the exploring robot, it was explored by sweeping its sensor to acquire a 3D scan and the mission was then considered completed.

We solved the coordination problem using the approach described in the disaster recovery application in Section 6. A sequence of actions that was planned in a particular situation can be seen in Fig. 16. Here, the passage was first blocked by an obstacle. The obstacle was then detected by the exploring robot. The coordination approach assigned the manipulation robot to clear the passage. After the passage was cleared and the manipulation robot moved to a waiting position, the exploring robot was able to reach the goal area and complete the task.

8 Discussion

In this paper we presented a novel technique to coordinate heterogeneous teams of exploring robots. We assumed that to explore the environment, the robots have to move in the environment and additionally are required to execute actions such as removing an obstacle or operating an elevator. These so-called symbolic actions stand in contrast to navigation actions that move robots to a given goal position in order to explore target locations. Our method integrates a symbolic temporal planning system and a robotic path planner. This combination allows our system to consider planning problems that include symbolic actions. To coordinate a team of robots, we generate a symbolic description of the current state of the system and of the goal state. These descriptions serve as input to the symbolic planner. To solve the coordination problem, the symbolic planner uses the path planner to efficiently estimate travel costs for the robots. In contrast to cost-based coordination approaches, our system is able to explicitly plan for the execution of symbolic actions. Still, the use of action costs that are determined by the robotic path planner allows us to generate time-efficient solutions. In this way, our approach combines the strength of cost-based coordination approaches with the flexibility of symbolic planning systems. By planning symbolic actions explicitly, the system is able to take into account actions such as manipulation actions or deployment and retrieval of robots.

The approach was implemented and evaluated extensively in simulated environments. We evaluated two distinct settings. First, we coordinated teams of marsupial robots that were able to deploy and pick up smaller robots. Second, we simulated a disaster scenario in which the task was to clear blockades and to perform pre-defined actions at certain critical locations in the environment. A similar setting was also investigated using a team of real robots.

The experimental results demonstrate that our approach can effectively coordinate large teams of robots and significantly outperforms handcrafted strategies. In addition to that, our planning framework adds a substantial degree of flexibility to the system. For example, additional constraints

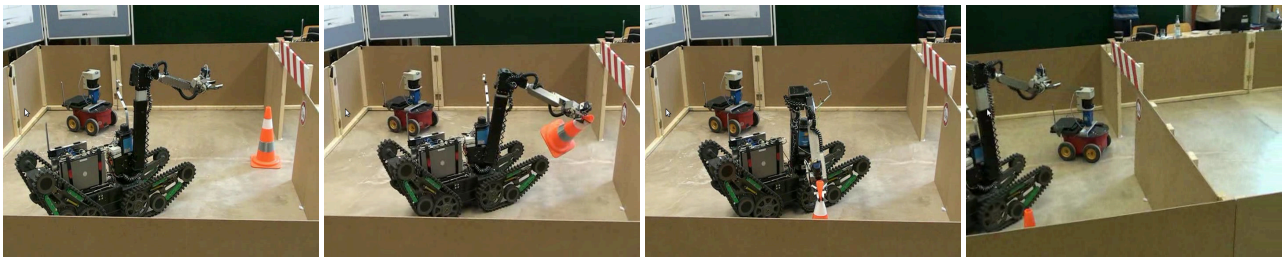

Fig. 16 Sequence of actions in real-world environment. The goal is to explore the area to the right of the robots. A narrow passage connects both areas but is blocked by an obstacle. From left to right: the obstacle is detected by the wheeled robot, the obstacle is removed by the manipulation robot, the manipulation robot returns to its initial position and the wheeled robot traverses the narrow passage to explore the target area.

such as power constraints for individual robots can be specified by adding adequate predicates to the problem description. Furthermore, other symbolic actions such as recharging batteries or deploying sensor nodes can be integrated in a straightforward way.

8.1 Limitations of the Approach

The planning system described in this paper generates temporal plans for the robots based on the current knowledge about the world. We do not assume a model of the unknown information and thus rely on an optimistic problem formulation. While the robots move, their state changes and new information about the environment is perceived. Therefore, we execute the planning cycle (illustrated in Fig. 2) in a continuous loop. The symbolic planer is implemented as an anytime algorithm that does not terminate after the first solution is generated. It produces a potentially non-optimal solution quickly and then improves upon this solution. If more than one solution is found, we set the planning timeout to 30 s. In the marsupial coordination setting, we evaluated our approach with up to 24 robots (6 carriers plus 18 rovers). For significantly larger teams, however, the problem complexity increases substantially and the solution reported by the anytime planner after 30 s may be suboptimal.

The exact problem size that can be solved close-to-optimally in a given planning time depends on the structure of the problem and on the team size. The evaluation of the planning time presented in Section 7.8 shows that the best plan will usually be found in less than 10 s when coordinating a team of five robots in a disaster recovery mission with up to ten mission targets. Many real-world scenarios will feature a similar complexity.

Acknowledgements We wish to thank Marc Gissler, Christoph Sprunk, and Matthias Westphal for their assistance during the real world experiments.

References

[2003] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt. Robot exploration with combinatorial auctions. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1957–1962, 2003.

[2001] B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1):5–33, 2001.

[2005] W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–378, 2005.

[1994] T. Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.

[2009] S. Cambon, R. Alami, and F. Gravot. A hybrid approach to intricate motion, manipulation and task planning. *International Journal of Robotics Research*, 28(1):104–126, 2009.

[1997] Y.U. Cao, A.S. Fukunaga, and A.B. Khang. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–27, 1997.

[2006] Y. Chen, B. W. Wah, and C.-W. Hsu. Temporal planning using subgoal partitioning and resolution in SGPlan. *Journal of Artificial Intelligence Research*, 26:323–369, 2006.

[2011] F. Cordes, I. Ahrns, S. Bartsch, T. Birnschein, A. Dettmann, S. Estable, S. Haase, J. Hilljegerdes, D. Koebel, S. Planthaber, et al. Lunares: lunar crater exploration with heterogeneous multi robot systems. *Intelligent Service Robotics*, pages 1–29, 2011.

[2002] F. Dellaert, T. Balch, M. Kaess, R. Ravichandran, F. Alegre, M. Berhault, R. McGuire, E. Merrill, L. Moshkina, and D. Walker. The Georgia Tech yellow jackets: A marsupial team for urban search and rescue. In *AAAI Mobile Robot Competition Workshop*, 2002.

[2009] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel. Semantic attachments for domain-independent planning systems. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pages 114–121, 2009.

[2007] A. Drenner and N. Papanikolopoulos. A Framework for Large-Scale Multi-Robot Teams. *Modeling and Control of Complex Systems*, page 297, 2007.

[1996] G. Dudek, M.. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4):375–397, 1996.

[1995] K. Erol, D. S. Nau, and V. S. Subrahmanian. Complexity, decidability and undecidability results for domain-independent planning. *Artificial Intelligence*, 76(1-2):75–88, 1995.

[2008] European Space Agency ESA. ESA's lunar robotics challenge website. http://www.esa.int/esaCP/SEMGAASHKHF_index_0.html, 2008.

[2009] P. Eyerich, R. Mattmüller, and G. Röger. Using the context-enhanced additive heuristic for temporal and numeric planning. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pages 130–137, 2009.

[2010] P. Eyerich, T. Keller, and M. Helmert. High-quality policies for the canadian traveler's problem. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, pages 51–58. AAAI Press, July 2010.

[1971] R. Fikes and N. Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[2003] M. Fox and D. Long. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research (JAIR)*, 20(1):61–124, 2003.

[1999] A. Gerevini and I. Serina. Fast planning through greedy action graphs. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 503–510, 1999.

[2008] A. Gerevini, A. Saetti, and I. Serina. An approach to efficient planning with numerical fluents and multi-criteria plan quality. *Artificial Intelligence.*, 172(8-9):899–944, 2008.

[2000] R. Grabowski, L.E. Navarro-Serment, C.J.J. Paredis, and P.K. Khosla. Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots*, 8(3):293–308, 2000.

[2012] P. Gregory, D. Long, M. Fox, and J.C. Beck. Planning modulo theories: Extending the planning paradigm. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2012.

[2008] M. Helmert and H. Geffner. Unifying the causal graph and additive heuristics. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2008.

[2002] M. Helmert. Decidability and undecidability results for planning with numerical state variables. In *Proc. of the Int. Conf. on Artificial Intelligence Planning and Scheduling*, pages 44–53, 2002.

[2006] M. Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[2001] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[2006] A. Howard, L.E. Parker, and G.S. Sukhatme. Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection. *International Journal of Robotics Research*, 25(5-6):447, 2006.

[2011] E.G. Jones, M.B. Dias, and A. Stentz. Time-extended multi-robot coordination for domains with intra-path constraints. *Autonomous robots*, 30(1):41–56, 2011.

[2003] E. Kadioglu and N. Papanikolopoulos. A method for transporting a team of miniature robots. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2297–2302, 2003.

[2011] L.P. Kaelbling and T. Lozano-Perez. Hierarchical task and motion planning in the now. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[1992] H. Kautz and B. Selman. Planning as satisfiability. In *Proc. of the European Conference on Artificial Intelligence (ECAI)*, 1992.

[2003] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3232–3238, 2003.

[2005] M. Koes, I. Nourbakhsh, and K. Sycara. Heterogeneous multirobot coordination with spatial and temporal constraints. In *Proc. of the National Conf. on Artificial Intelligence*, page 1292, 2005.

[1999] R.R. Murphy, M. Ausmus, M. Bugajska, T. Ellis, T. Johnson, N. Kelley, J. Kiefer, and L. Pollock. Marsupial-like mobile robot societies. In *Proc. of the Annual Conf. on Autonomous Agents*, page 365, 1999.

[1991] C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150, 1991.

[2010] S. Richter and M. Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research (JAIR)*, 39:127–177, 2010.

[2007] J. Rintanen. Complexity of concurrent temporal planning. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2007.

[2010] S.J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Prentice hall, 2010.

[2000] P.E. Rybski, S.A. Stoeter, M.D. Erickson, M. Gini, D.F. Hougen, and N. Papanikolopoulos. A team of robotic agents for surveillance. In *Proc. of the Int. Conf. on Autonomous Agents*, pages 9–16, 2000.

[1993] K. Singh and K. Fujimura. Map making by cooperating mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 254–259, Atlanta, GA, USA, 1993.

[2009] C. Stachniss, O. Martinez Mozos, and W. Burgard. Efficient exploration of unknown indoor environments using a team of mobile robots. *Annals of Mathematics and Artificial Intelligence*, 52:205ff, 2009.

[2009] C. Stachniss. *Robotic Mapping and Exploration*, volume 55 of *STAR Springer tracts in advanced robotics*. Springer, 2009.

[2008] K.M. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

[2009] K.M. Wurm, R. Kümmerle, C. Stachniss, and W. Burgard. Improving robot navigation in structured outdoor environments by identifying vegetation from laser data. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[2010] K.M. Wurm, C. Dornhege, P. Eyerich, C. Stachniss, B. Nebel, and W. Burgard. Coordinated exploration with marsupial teams of robots using temporal symbolic planning. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, October 2010.

[1997] B. Yamauchi. A frontier based approach for autonomous exploration. In *Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 146–151, 1997.

[1998] B. Yamauchi. Frontier-based exploration using multiple robots. In *Proc. of the Int. Conf. on Autonomous Agents*, pages 47–53, 1998.

[2002] R. Zlot, A.T. Stenz, M.B. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2002.