Empirical Study of Planning and Execution for Large Teams of Robots

Daniel Saur, Tareq Razaul Haque, and Kurt Geihs

Distributed Systems Group, University of Kassel 34121, Germany
{saur,haque,geihs}@vs.uni-kassel.de

Abstract. Large teams of robots can substantially increase the effectiveness of planning by acting as coordinated team. Our focus is on the planning of activities of a team of autonomous, mobile robots by distributed planning coordinated by one robot. With arising number of agents the communication increases rapidly. Our goal is to minimize communication much as possible. Modeling needs to be combined with planning to describe complex activities in intuitive way. The main contribution of the paper is the optimization of the planning process while using every agent as a planning resource and aiming at low communication needs. We evaluated our distributed planning for teams of up to 75 agents in the transport domain of the International Planning Competition[1] (IPC). We optimized the planning process compared to state-of-the-art approaches (last winners of IPC in the transportation domain) by up to 23%.

Keywords: Autonomous robots, Mobile robots, Distributed planning

1 Introduction

Recent advances in autonomous robot technology have opened up great opportunities in an exciting new application potential. Autonomous mobile robots can act individually while using an intuitive goal description for the team. This creates an enormous potential for innovative applications that intelligently support environmental monitoring, disaster management, logistics operations, and many other practices.

However, several challenging research questions have to be solved before we can harvest the benefits of such kinds of multi-agent systems. Increasing the number of agents also enourmously increases the overhead for maintenance, modeling, and testing. In our work we concentrate on planning for large teams with a high number of agents. The planning process calculates a plan, which describes the activities of all agents from a global perspective. This plan is divided into tasks, which denote the activity of a specific agent within the plan. The plan format is defined with ALICA (A language for interactive cooperative Agents)[15], which offers support for task allocation and coordination. Finally, we use every agent

[1] http://ipc.icaps-conference.org/

as a planning resource. The goal is to optimize the search time in distributing different seeds for the search tree, where we expect an acceleration of the search. We report on the results of an ongoing research project where we are developing a framework which supports distributed planning for a team of robots. We accelerated the search time by as much as 23% for autonomous mobile robot teams, consisting of as many as 75 agents, using a linearly scalable communication of agents.

The reminder of this paper is organized as follows. In the next section we outline the requirements of multi-agent planning for teams of up to 75 agents. In section 3, we start to discuss related works. In section 4, we introduce the basics of ALICA, which is used to describe team activities/plans. Furthermore, we sketch the basics of the planning framework pRoPhEt MAS [13]. Finally, in section 5 we evaluate the planning framework on relevant scenarios from the International Planning Competition.

2 Requirements

The requirements for multi-agent teams with a high number of agents in a team are:

- Modeling the global and local activities
- Automatic plan creation
- Low communication overhead

The description of team activities for autonomous mobile robots requires a suitable and intuitive description, instead of providing only single agent programs [15]. Furthermore, we would like to support task allocation and coordination instead of using predefined task-specific mapping to certain robots. With an increasing number of agents, manual modeling and task mapping takes a lot of time, and is hard to maintain. Hence, multi-agent systems require an intuitive method to control robot activities, and one that offers easy integration.

The communication bandwidth is limited. Hence, the planning process must also aim at keeping the number and size of messages low, particularly for large agent teams.

Finally, describing activities of autonomous mobile robot teams with an increasing number of agents requires a combination of modeling and planning.

3 Related Work

Heuristic search has become the predominant feature of problem solving for several years. The Fast Downward Planner [7] is a classical planning system based on heuristic search. Fast Downward is a best-first search planner that utilizes the information from domain transition graph as the heuristic to guide the search. Thus, it can deal with general deterministic planning problems encoded in the propositional fragment of PDDL2.2. The basic idea for the development of PDDL

[6] was to define a common interface to describe this problem class. PDDL defines a language to describe the existing world, actions to execute by agents and the goal state. The International Planning Competition (IPC) takes place every year, where newly developed planners evaluate difficult planning problems.

Helmert et al. [8] have proposed a concrete strategy for abstraction to derive better heuristics, and have empirically demonstrated the power of the merge-and-shrink abstraction heuristics. In particular, the empirical evaluation of the merge-and-shrink abstractions by Helmert et al. [8] suggests that, for many tasks, using a set of abstractions improves the overall heuristic guidance.

Brenner and Ivana [3] presented a new algorithmic framework in which situated dialogue is modeled as Continual Collaborative Planning (CCP). They showed how mixed-initiative dialogue that interleaves physical actions, sensing, and communication between agents occurs naturally during CCP. Thus, they introduced the language MAPL [4]. Their article describes a continual planning algorithm realized with MAPL. For the proof-of-concept, Brenner and Nebel evaluated MAPL in the grid world domain, where a team of four robots must find their position in the grid.

HPLAN-P [1] performs forward search using heuristics designed for propositional preferences. These are based on the relaxed planning graph (RPG) structure and use techniques such as summing the layers in which goals/preference facts appear (rather than relaxed plans) to estimate goal distance and preference satisfaction potential.

LAMA [11] is a classical planning system based on heuristic forward search. The system uses two heuristic functions in a multi-heuristic state-space search: a cost-sensitive version of the FF heuristic, and a landmark heuristic guiding the search towards states where many subgoals have already been achieved. Action costs are employed by the heuristic functions to guide the search to cheap goals rather than close goals, and iterative search improves solution quality while there is time remaining.

Burns et al. [5] developed parallel versions of best-first search to harness modern multicore machines. They showed that a set of previously proposed algorithms for parallel best-first search can be much slower than running A* sequentially. They presented a hashing function for parallel retractin A* (PRA*) that takes advantage of the locality of a search space and gives superior performance. They also presented another algorithm, PBNF, which approximates a best-first search ordering while trying to keep all threads busy.

Nissim et al. [9] developed a distributed planning system which uses a heuristic forward search. This system is evaluated for different IPC problems. The main disadvantage to this system is that communication effort increases rapidly as the number of agents increases.

The main contribution of most of state-of-the-art planning systems is to optimize the search heuristic. However, the quality of the search heuristic depends on the test domain. Our focus is to optimize planning independent of the search heuristic. Distributed planning often relies on high communication as in [9]. In

real world applications like RoboCup[2] low communication approaches are required [14].

4 Planning Framework

In this section, we briefly introduce the planning framework. We will first introduce the basics of ALICA [15], and then we will sketch the basics of pRoPhEt MAS [13].

4.1 ALICA

ALICA is a language for describing team activities of interactive mobile agents from a global perspective. Originally, it was developed for the RoboCup Middle Size League. However, it has also been shown to be a viable and effective solution for other application domains, such as exploration robots [12] and autonomous vehicles in traffic [10].

The core elements of the language [14] are shown in Table 1.

$(\mathcal{A}, \mathcal{L})$	the domain signature	The domain signature consists of the set of possibly interacting agents and the logic with which the world is represented.
$\mathcal{R}$	a set of roles	This set contains all availables roles any agent can be assigned to.
$\mathcal{B}$	a set of behaviours	Behaviours are atomic action programs that form the means to interact with the environment.
$\mathcal{P}$	a set of plans	Each plan describes a specific cooperative activity.
$\mathcal{P}_\vee$	a set of plantypes	A plantype is a set of alternative plans.
$\mathcal{O}$	a set of planning problems	Defines a goal condition and a set of $\mathcal{P}$, to achieve the goal condition.
$\mathcal{T}$	a set of tasks	Each task intuitively describes a function or duty within plans, meant to be fulfilled by one or more agents.
$\mathcal{Z}$	a set of states	A state occurs within a plan as a step during an activity. It can contain plantypes and behaviours.
$\mathcal{W}$	a set of transitions	Each transition (z_1, z_2, ϕ) relates a predecessor state z_1 with a successor state z_2 and a condition $\phi \in \mathcal{L}(Pred, Func)$.

Table 1. Elements of a ALICA Program

The individual logic elements $\mathcal{L}$ defined by $\mathcal{L}(Pred, Func)$ are structured using the functions listed in Table 2.

[2] http://www.robocup.org

States: $\mathcal{P} \mapsto 2^{\mathcal{Z}}$	States maps plans to the set of contained states.
Tasks: $\mathcal{P} \mapsto 2^{\mathcal{T}}$	Tasks maps plans to the set of related tasks.
ξ: $\mathcal{P} \times \mathcal{T} \mapsto \mathbb{N}_0 \times (\mathbb{N}_0 \cup \{\infty\})$	ξ defines the upper and lower bound of agents assignable to a task τ in plan p.
Pre: $\mathcal{P} \cup \mathcal{B} \mapsto \mathcal{L}_S$	Pre(p) denotes the precondition of plan or behaviour p.
Run: $\mathcal{P} \cup \mathcal{B} \mapsto \mathcal{L}_S$	Run(p) denotes the runtime condition of plan or behaviour p.
PlanTypes: $\mathcal{Z} \mapsto 2^{\mathcal{P}_\vee}$ Behaviours: $\mathcal{Z} \mapsto 2^{\mathcal{B}}$	PlanTypes(z) denotes the set of plantypes to be executed in state z. Behaviours(z) denotes the set of behaviours to be executed in state z.
Post: $\mathcal{Z} \mapsto \mathcal{L}_S$	Post(z) is a partial function, that maps terminal states of a plan to postconditions.
$\mathcal{U}$: $\mathcal{P} \mapsto 2^{\mathcal{L}_S} \mapsto \mathbb{R}$	$\mathcal{U}(p)$ is the utility function of p, evaluating p with respect to a set of formula.

Table 2. Structure Definitions of a ALICA Program

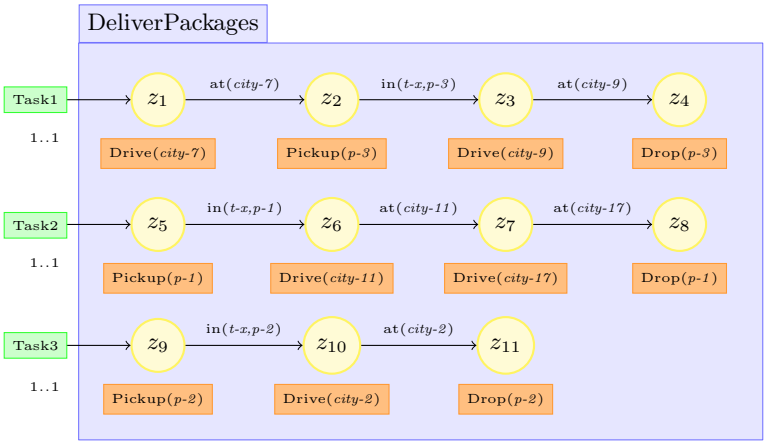

Fig. 1. Example ALICA plan for delivering packages by multiple agents

Figure 1 shows an example ALICA plan using the core elements of the language. This figure shows an example from the transport domain[3]. We defined roles $\mathcal{R}$ that are suitable for the task $\mathcal{T}$ dependent on the robot capabilities. Every agent in the team can assign to one of the tasks with respect to the minimum and maximum cardinalities (ξ) 1..1. The "DeliverPackages" plan $\mathcal{P}$ contains a state machine for every agent in team with several states $\mathcal{Z}$. Every state machine contains a plan, which in turn contains a state machine of basic behaviours $\mathcal{B}$. These plans represent the basic skills from the transportation domain. The basic skills of the agents are "Pickup", "Putdown" and "Drive". The agents can switch states with conditional transitions. The plan realizes the delivery of three packages.

In order to model plans ALICA offers a "PlanDesigner" which is a graphical tool based on the Eclipse Development Platform [2]. It supports modelling of all parts of an ALICA program, i.e., roles, tasks, plans, plantypes, utility functions, and conditions, as well as generating code from the models in a model-driven development fashion. The Ecore model is shown in figure 2. Modelled plans are stored in the XMI format and loaded afterwards by the runtime engine. However, for efficiency reasons, the tool provides mechanisms for generating platform-specific code for the evaluation of conditions and utilities. Since these evaluations happen very frequently during runtime, the generation of platform-specific code, which can be executed directly, results in enormous efficiency benefits. In order to facilitate an intuitive understanding, language elements are represented graphically.

4.2 pRoPhEt MAS

The planning framework pRoPhEt MAS (Reactive Planning Engine for Multi Agent Systems) is divided into two major parts (see Figure 3). The first part consists of "World" and "ALICA-Engine" and represents the basic ALICA components. The "ALICA-Engine" is the implementation of the language elements for section 4.1. In addition ALICA offers further algorithms for task allocation, role-task-mapping, supports coordinated execution in dynamic environments [14]. The "PlanBase" contains all modeled ALICA-plans that the team can access. Dependent on the actual world situation, ALICA will then select a suitable plan while reacting quickly to world changes. The second part consists of "ISharing" and "IPlanner". These components are used to expand the basics ALICA by a planning engine. "ISharing" is used to communicate plans after creation, and electing a leader, which starts the planning process. The election criteria can be defined by implementing the ISharing interface. At this time the robot with lowest id will be leader.

If the "PlanSelector" selects a plan containing a planning problem $\mathcal{O}$ (see language elements of section 4.1), which is briefly defined by basic actions and a goal description, the leader will start the planning process by the "PlannerBase". The resulting plan from the "PlannerRealization" will be communicated to all

[3] http://ipc.icaps-conference.org (IPC 2011)

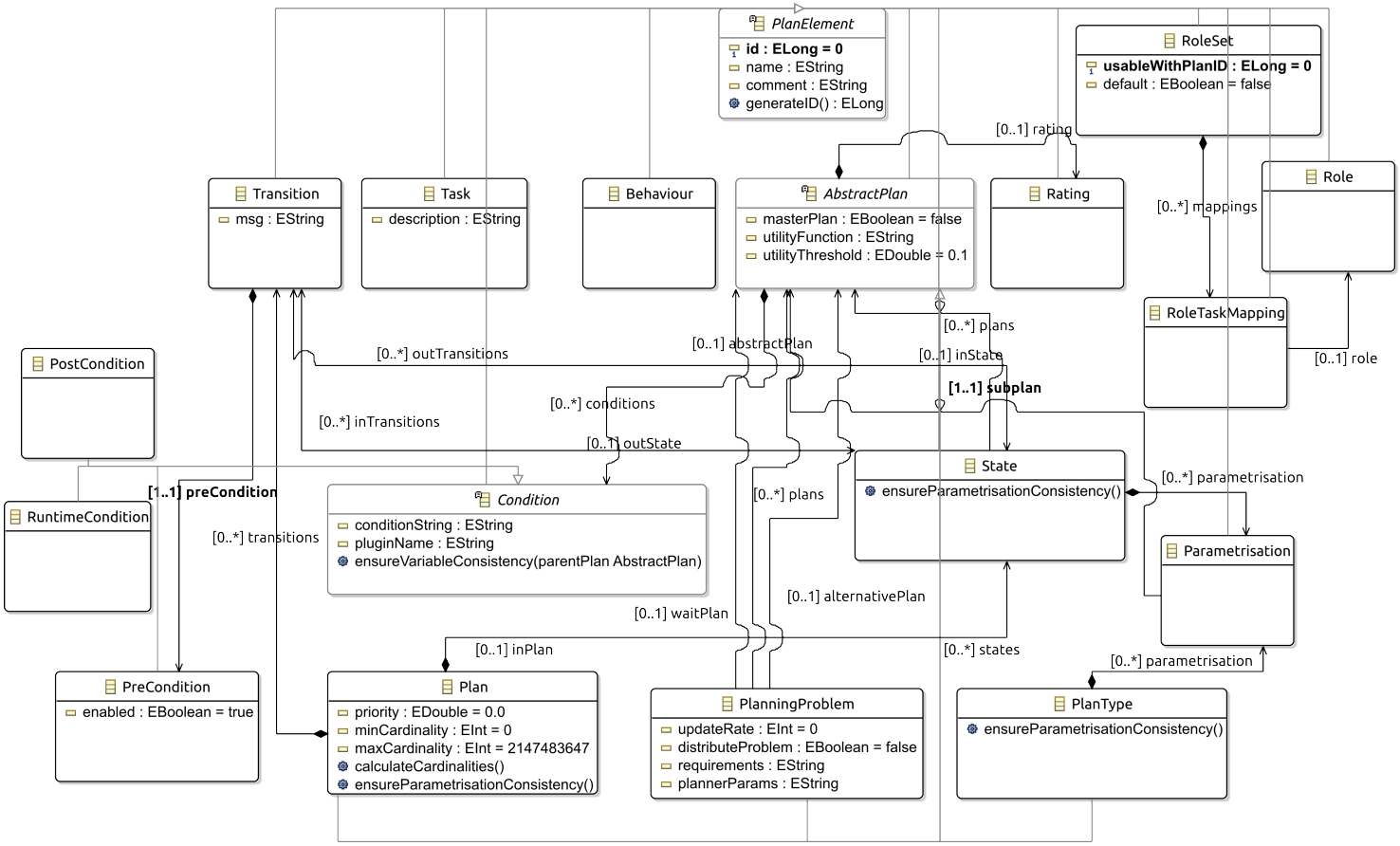

Fig. 2. Ecore model of ALICA's basic elements

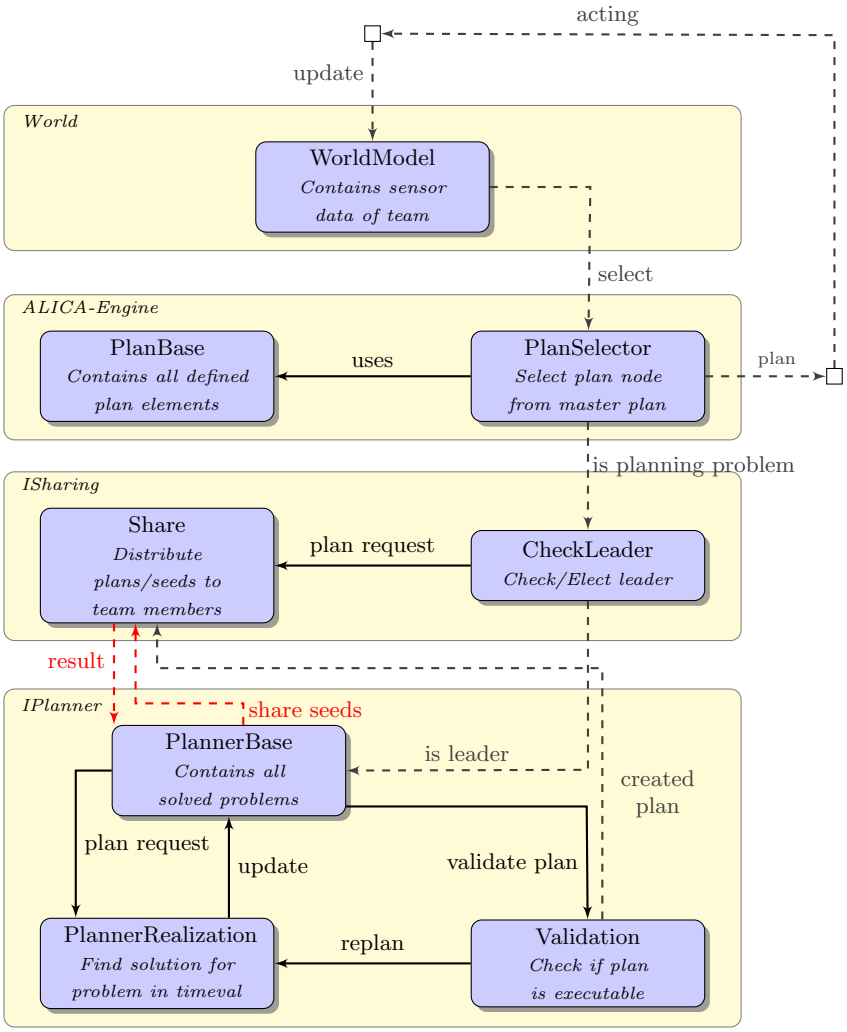

Fig. 3. Planning framework consisting of ALICA for describing team activities expanded by a planning engine

agents, if this plan is validated correctly by the "Validation" component. Hence, ALICA can react quickly in dynamic environments as evaluated in real world scenarios [13], though this is not the focus of this paper.

In order to decrease the search time and save memory for the planning process, the leader is able to distribute seeds of the search space to teammates using the "PlannerBase", which is shown in Figure 4. If an agent receives a seed, it will start the search, and send the solution back to the leader. If the leader receives the first result, he will share this solution to all other members, which will stop the search.

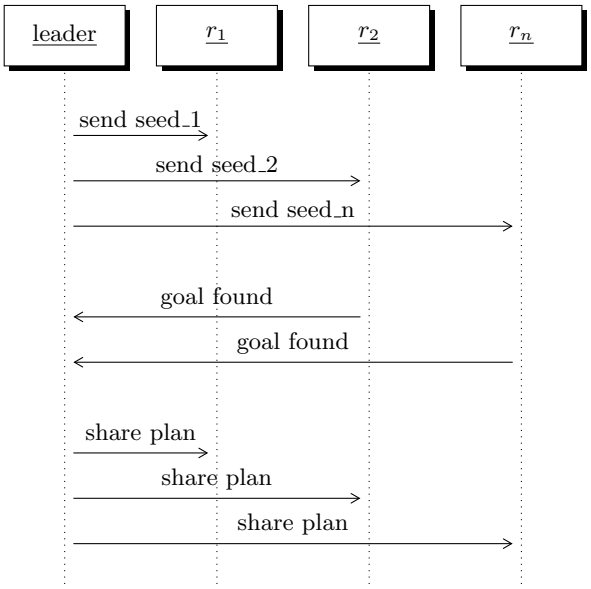

Fig. 4. Share seeds to accelerate search

5 Evaluation

In order to evaluate our framework, we use the defined problem of IPC 2011, as described in Section 3, and compare our results to state-of-the-art approaches. We used the planner "seq-sat-fdss-1", based on a Fast Downward Planning System [7], which we modifed to allow seed sharing with the entire team. The planner "seq-sat-fdss-1"participated in IPC 2011 and came in 2nd place in the transportion domain. Experiments were run on an Intel i7-2630QM CPU 2.00GHz processor, where we were allowed to use maximum one core. The results are shown in Figure 5, which shows calculation time and costs for different problems. These problems differ in map size, number of agents and packages to deliver, and can found on the IPC website. The time shows the total search time.

In the transportation domain the costs are defined by the total traveled distance. On average our approach decreases the costs by 1.3%. In addition, we were able to reduce the calculation time on average by 28.3%. For Problem 14, we reduced the calculation time by 65%.

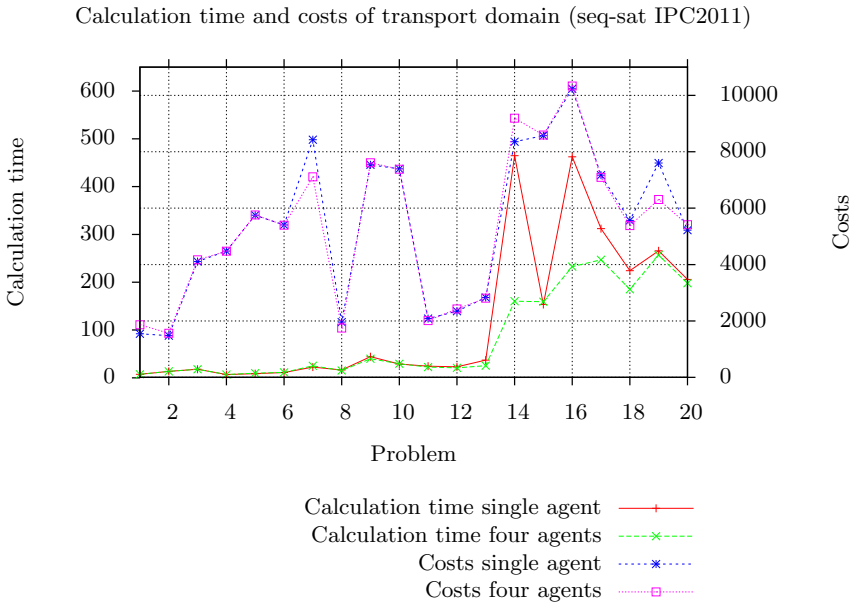

Fig. 5. Calculation time and costs for transport (IPC2011)

We later extended the problem and created a random map with 100 locations as shown in Figure 6. A distributed team with n members has to deliver n packages. Imagine a mail service group in a city that has around 100 different locations. This mail service wants to exchange packages between these locations via mobile, autonomous agents like copters or cars. The problem is how the agents should be assigned to deliver all packages. The results are shown in Figure 7. In this Figure the x represents the number of agents and the y shows calculation time and costs. The costs increased on average by 0.01%, but the execution time decreases on average by 10.17%.

The distributed planning scales linearly as $3 * (n - 1)$ with the number of agents. In a first round, we distribute seeds to all team members, which takes $(n - 1)$ messages. Next, in the worst case we wait for $(n - 1)$ results. Finally, we distribute the result to all $(n - 1)$ members. After receiving the result, every robot will stop the search.

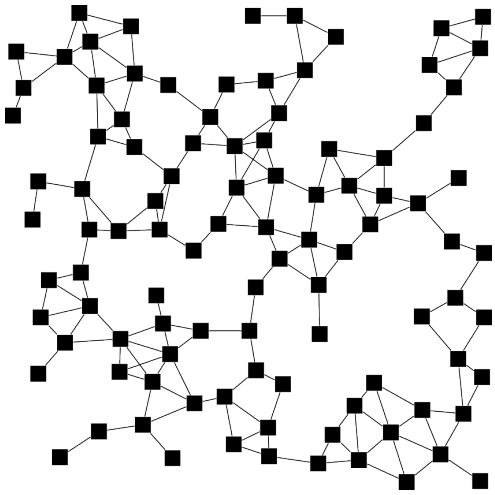

Fig. 6. Example map of transportation scenario

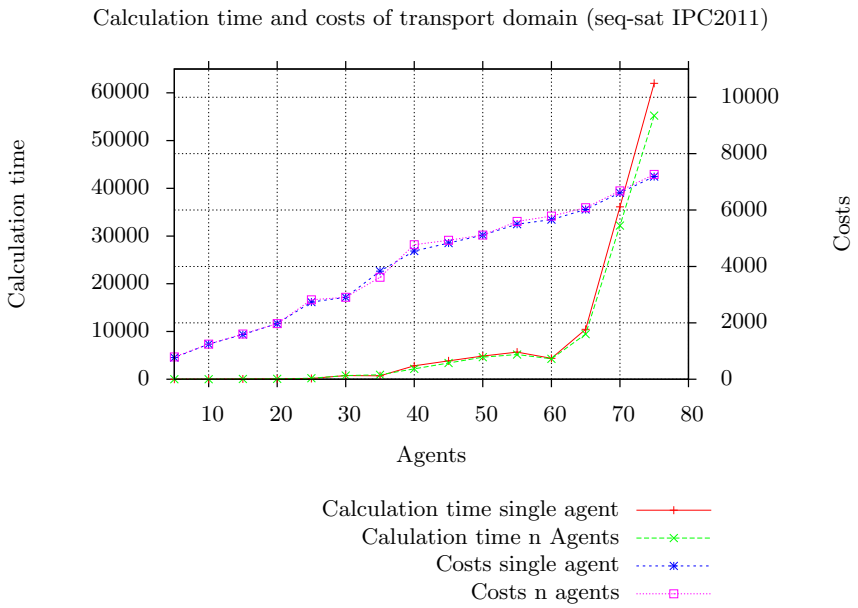

Fig. 7. Calculation time and costs of the map in Figure 6

6 Conclusions

The task planning for teams with a large number of mobile autonomous robots still offers improvements in research. The major problem is that cooperative distributed planning increases the communication rapidly as the number of agents increases. On the other side, severe resource limitations apply to the strategy of central planning, if complex planning problems shall be dealt with. Hence, it creates an opportunity to optimize planning for scenarios like disaster management, logistics operations, and many more.

However, planning is an increasingly complex task in multi-agent systems for an increasing number of robots. The state space grows tremendously with the number of robots. Moreover, in such domains, automatic plan generation reduces the overhead for maintenance, modeling, and testing enormously. Thus, planning is an important part of describing the activities of multi-agent systems.

The strategy of our framework is to *divide and conquer* to cope planning problems regarding resources like memory and communication bandwidth. Therefore, we use all robots as planning resources to reduce the planning time and divide the memory usage. The found solution of the robots will be shared, intermediately. Moreover, the communication burden scales linearly with an increasing number of agents.

In our evaluation, we took the transport scenario of the IPC2011 to compare our planning system to the state-of-the-art planner. Furthermore, we created more complex scenarios for the transport domain with up to 75 agents. We were able to improve the search time by up to 65% and 19.3% on average in the IPC problems. The costs in both scenarios were nearly the same (1.3% difference).

Our next steps are to evaluate the framework in the RoboCup domain using additional computational units to realize a set play in this dynamic environment.

References

1. Jorge A. Baier, Fahiem Bacchus, and Sheila A. McIlraith. A heuristic search approach to planning with temporally extended preferences. *Artificial Intelligence*, 173(5-6):593–618, 2009.
2. W. Beaton and J. d. Rivieres. Eclipse Platform Technical Overview. Technical report, The Eclipse Foundation, 2006.
3. M. Brenner and I. Kruijff-Korbayov. A Continual Multiagent Planning Approach to Situated Dialogue. In *Proceedings of the LONDIAL (The 12th SEMDIAL Workshop on Semantics and Pragmatics of Dialogue)*. LONDIAL, 6 2008.
4. Michael Brenner and Bernhard Nebel. Continual planning and acting in dynamic multiagent environments. *Autonomous Agents and Multi-Agent Systems*, 19(3):297–331, June 2009.
5. Ethan Burns, Sofia Lemons, Wheeler Ruml, and Rong Zhou. Best-first heuristic search for multicore machines. *Journal of Artificial Intelligence Research*, 39(1):689–743, 2010.
6. M. Ghallab, C. K. Isi, S. Penberthy, D. E. Smith, Y. Sun, and D. Weld. PDDL - The Planning Domain Definition Language. Technical report, CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

7. M. Helmert. *The Fast Downward Planning System*. Journal of Artificial Intelligence Research 26, 2006.
8. Malte Helmert, Patrik Haslum, and Jrg Hoffmann. Flexible abstraction heuristics for optimal sequential planning. In *ICAPS*, pages 176–183, 2007.
9. R. Nissim and R. I. Brafman. Distributed Heuristic Forward Search for Multi-Agent Systems. *Computing Research Repository (CoRR)*, abs/1306.5858, 2013.
10. Stephan Opfer, Andreas Witsch, and Kurt Geihs. A Formal Multi-Agent Language for Cooperative Autonomous Driving Scenarios. nov 2014.
11. Silvia Richter and Matthias Westphal. LAMA is a classical planning system based on heuristic forward search. *Journal of Artificial Intelligence Research*, (39):127177, 2010.
12. D. Saur, T. R. Haque, R. Herzog, and K. Geihs. MAGiC : Multi-Agent Planning using Grid Computing concepts. In *12th International Symposium on Artificial Intelligence, Robotics and Automation in Space - i-SAIRAS 2014*, Quebec Canada, 2014.
13. Daniel Saur and Kurt Geihs. pRoPhEt MAS: Reactive Planning Engine For Multi-agent systems. In *13th International Conference on Intelligent Autonomous Systems*, 2014.
14. H. Skubch. *Modelling and Controlling of Behaviour for Autonomous Mobile Robots*. Westdeutscher Verlag GmbH, 2013.
15. H. Skubch, M. Wagner, R. Reichle, and K. Geihs. A modelling language for cooperative plans in highly dynamic domains. *Mechatronics*, 21:423–433, 2011.