MAGiC : Multi-Agent Planning using Grid Computing concepts

D. Saur, T. R. Haque, R. Herzog, K. Geihs

*Distributed Systems, University of Kassel, Germany
e-mail: last_name@vs.uni-kassel.de

Abstract

This paper presents a distributed planning approach which is able to produce a plan with minimum communication among the agents of the team. We developed a fusion planner which considers the limitations of distributed and centralized planning. Principally, a central agent distributes tasks among multiple agents to solve a problem in reduced search space compared to centralized planner. Moreover, the designed planner requires lesser communication than a distributed planner among multiple agents. We perform evaluation on NASA's Mars Rover domain and compare the proposed planner with similar ones. A general conclusion is that the planning process for a team of agents can be improved compared to the state of the art approaches by up to 46.4% faster and 98.2% lesser messages.

1 Introduction

In many situations, we experience planning and decision making problems like an air traffic controller has to assign slots for takeoff and landing, or a shipping and logistics company has packages to deliver all over the world and more. These problems are multi-agent problems, for which centralized or distributed approaches exist. However, none of them is perfect and both have its drawbacks, e.g. a centralized planner faces the memory limitation and task overloading problem as it puts entire computational burden to a single agent [1]. Thus, agent failure can raise serious robustness issues. To solve these problems researchers developed distributed planner. A single problem is split into multiple tasks and distributed across multiple agents. Distributed planning indeed improves the performance by reducing computation time but it requires more frequent communication among agents [1].

After consideration of the limitations that each approach has, we have designed a planner based on the Fast Downward Planner [2] which finds solutions in a team-oriented fashion. A central agent divides the total problem search space according to the available agents using BOINC [3]. BOINC is an open source middleware system for volunteer and grid computing. We configured BOINC for a local network, and every agent receives only one task. Hence, each agent has a smaller search space to compute the solution. Moreover, agents only have to communicate with central agent while receiving the task and after finding a solution. Therefore, communication scales linear up to the number of agents. This approach is robust to agent failure, because the central agent assigns the task to another teammate to find the solution.

The reminder of this paper is organized as follows. At beginning we outline in section 2 the requirements of MAGiC. In section 3 we start to discuss related works. Next, we introduce the basics of ALICA in section 4, which offers the basis for describing team activities (plans). ALICA allows the modeling of behaviors for a team from a global perspective instead of interfacing single agent programs. Finally, we sketch in the planning framework pRoPhEt MAS [5], which is expanded by MAGiC. Moreover, we used NASA's Mars Rover domain to evaluate the planner with the expansion of MAGiC shown in section 5, followed by the conclusion in section 6.

2 Requirements

The requirements for multi-agent planning for teams up to 20 agents with low communication bandwidth:

- Modeling the global and local activities
- Ease of use
- Optimize planning time and costs
- Low communication overhead

Working with multiple autonomous mobile robots requires a suitable, human readable, intuitive description of team activities from a global perspective instead of providing only single agent programs [13]. Moreover, we would like a common support for coordination of the team without a predefined specific task mapping to certain robots.

If the number of agents of a team increases, manual integration takes a lot of time, and is hard to maintain and evolve. Hence, multi-agent systems require an intuitive method to control robot activities, which offers easy integration.

Divide the planning to all member of the team should decrease the search time and minimize the costs (action steps).

The communication bandwidth is rather low. Therefore, the planning process must aim at keeping the number of messages low, particularly for larger agent teams.

Combining state of the art methods of modeling and planning is required, to handle the process to describe and plan the activities of autonomous mobile robots.

3 Related Work

Brenner and Nebel introduced the language MAPL [4]. The article describes a continual planning algorithm realized with MAPL. For the proof-of-concept, Brenner and Nebel evaluated MAPL in the grid world domain. They employed a small team consisting of four robots to find their position in the grid.

Nissim et al. [1] developed a distributed planning system which uses a heuristic forward search. This framework is evaluated for different IPC problems. The main disadvantage is that the communication effort increases rapidly as the number of agents increases.

The basic idea for the development of PDDL [6] was to define a common interface to describe planning problems. 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 hard planning problems. At competition the organizer presents a couple of planning problems defined in PDDL, where every participating planner has to solve these problems.

The Fast Downward Planning System [2] is a classical planning system based on heuristic search, encoded in the propositional fragment of "PDDL2.2" and developed by Helmert. It can deal with general deterministic planning problems.

G. Antonelli et al. [7] offers an approach to formation control with collision avoidance of a multi-robot system. It is a hierarchy-based approach that uses null-space projection to handle eventually conflicting tasks.

Tuci et al. presents [8] methods to cope with cooperative transport tasks. The first experiment is a study on the utility of self-assembling robots to address relatively complex scenarios. The second experiment is an attempt to enhance the adaptiveness of multi-robot system.

4 Planning Framework

In this section we briefly introduce ALICA (A language for interactive cooperative Agents) [9] which is the foundation for our planning solution. Finally, we present the planning framework which uses MAGiC to optimize the planning process.

4.1 ALICA

ALICA provides modeling facilities for cooperative agent behaviours with clear operational semantics. Originally, it was developed for our football robot team in the RoboCup Middle Size League[1]. Later it has been shown to be a viable and effective solution for other application domains as well, such as exploration robots, autonomous vehicles in traffic.

The core elements of the language are:

- Basics:
  - $\mathcal{L}$: Language $\mathcal{L}(Pred, Func)$ describes the agents' belief with a set of predicates $Pred$ and a set of function symbols $Func$.
  - $\mathcal{R}$: Set of roles, which the agent can take.
  - $\mathcal{B}$: Set of atomic behaviours, which can interact with the environment.
  - $\mathcal{P}$: Set of cooperative behavior description.
  - $\mathcal{O}$: Defines a goal condition and a set of $\mathcal{P}$, to achieve the goal condition.
  - $\mathcal{T}$: Every task describes a function in a plan.
  - $\mathcal{Z}$: States exist in plans and represent a step in this plan.
  - $\mathcal{U}$: A utility function defines quality of $\mathcal{P}$ depends on role and task mapping.
- Conditional elements:
  - Pre - denotes a pre-condition of a behavior or plan defined by Pre: $\mathcal{P} \cup \mathcal{B} \mapsto \mathcal{L}$.
  - Run - denotes a runtime-condition of a behavior or plan defined by Run: $\mathcal{P} \cup \mathcal{B} \mapsto \mathcal{L}$.
  - Post - denotes a post-condition of a behavior or plan defined by Post: $\mathcal{Z} \mapsto \mathcal{L}$.

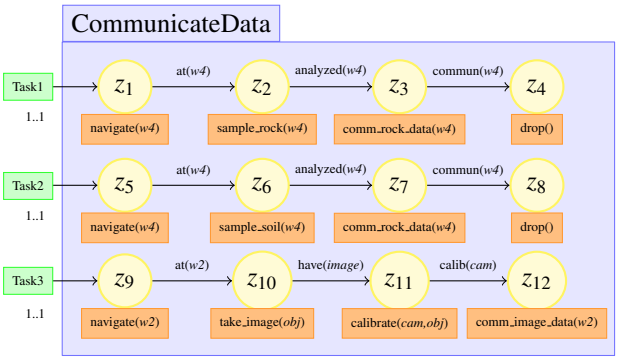

Figure 1. Example ALICA plan

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

Figure 1 shows an example ALICA plan using the core elements of the language. This figure shows an example of the rovers' domain[2]. In this problem, rovers must carry out plans that make optimal use of their resources for collecting soil and rock samples, and communicating back to the lander. This problem is quite similar to the IMPERA [10] domain, a team of robots has to explore the environment to find stones and carry them to certain location for building specific structures. We defined roles $\mathcal{R}$ which 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 1..1. The "CommunicateData" plan $\mathcal{P}$ contains a state machine for every agent in team with several states $\mathcal{Z}$. Every state contains a plan, which contains a state machine of basic behaviors $\mathcal{B}$. These plans represent the basic skills of the rovers domain . The agents can move to next states with conditional transitions. In the plan a team of robots have to communicate rock- and soil-data of waypoint four, and communicate image-data from waypoint two.

4.2 From pRoPhEt MAS towards MAGiC

We begin by explaining how to use planning problems $\mathcal{O}$ for a team of autonomous mobile agents. Figure 2 shows an example for the rovers domain. At the beginning it is possible to start and stop the team's behavior. This plan illustrates that the team of agents have to localize first. Every agent can sample objects only, if he is localized, and interesting waypoints are known. We explicitly modeled a plan to solve the exploration. The "CommunicateData" is defined as the planning problem, which may contain basic skills ($\mathcal{P}$ defined in $\mathcal{O}$) like "navigate", "sample_rock", "sample_soild", "sample_rock" etc.; therefore, the engine tries to find a plan at runtime. The engine offers two modes to solve planning problems:

- Online : Before defined goal will solved automatically.
- Interactive : User sends goal description interactively.

If the goal is defined at beginning, the problem will be solved automatically, when robots enter the plan "CommunicateData". But often the problem definition depends on former team behaviors, the environment and so on. For this reason the user is able to set a goal interactively.

The planning engine, pRoPhEt MAS [5], searches a valid sequence of actions to satisfy the goal. The coordination execution of the resulting plan is handled by ALICA. Every root plan (Top-Level) must span an acyclic "plan tree" like in Figure 2. In every part of the "plan tree" planning problems $\mathcal{O}$ can be used.

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

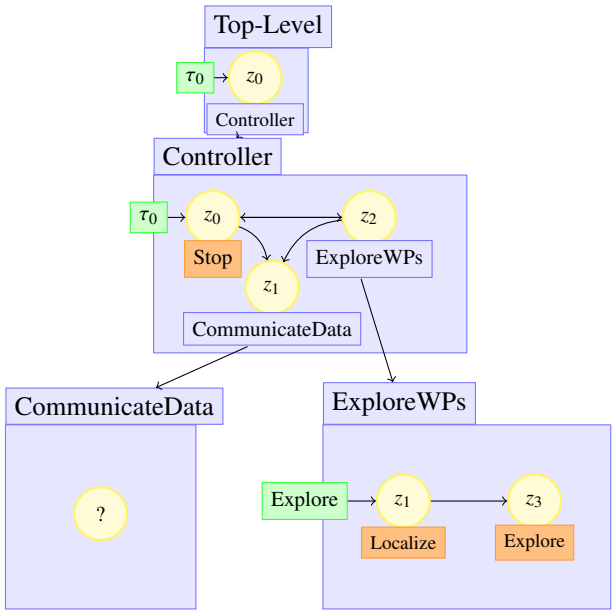

Figure 2. Example plan for usage of planning problems

The planning framework pRoPhEt MAS is shown in Figure 3. "World" and "ALICA-Engine" are the basic components of the framework. The "WorldModel" contains the information about the robots' environment. ALICA selects a suitable plan from the "PlanBase" which is modeled by the developer. ALICA reacts on environment changes, and reactively selects new plans. If a planning problem $\mathcal{O}$ is defined, the engine will use a planning system for creating a suitable plan. (Of course, planning problems can be solved offline as well.) The plan generation is handled by a central leader to reduce the amount of communication. The leader is selected by the team implicitly, while sharing team information periodically. The robot with the lowest ID becomes the leader. After finding a solution, the leader shares the plan with all members after the plan has been validated by the "Validation" component.

The expansion of MAGiC into pRoPhEt MAS is shown in 3. Every "PlanBase" contains a BOINC server. Now, every agent is able to offer task for the search process. The leader starts to divide the search problem into tasks with different search seeds depending on the number of agents in the team. The principle operation is shown in 4. The leader shares seeds in order to decrease the search time and save memory. If an agent receives a seed, he 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.

ALICA shares the team information every 100ms. Therefore, every agent knows which agents are member

of the team. If the leader breaks down while solving a planning problem, the team will stop the search and elect a new leader. If a team member breaks down and the planning problem is not solved, the leader will send the non solved seed to a team member, which solved yet a seed.

MAGiC scales linear $3*(n-1)$ which increasing number of agents. Therefore, MAGiC can be used up to a high number of agents.

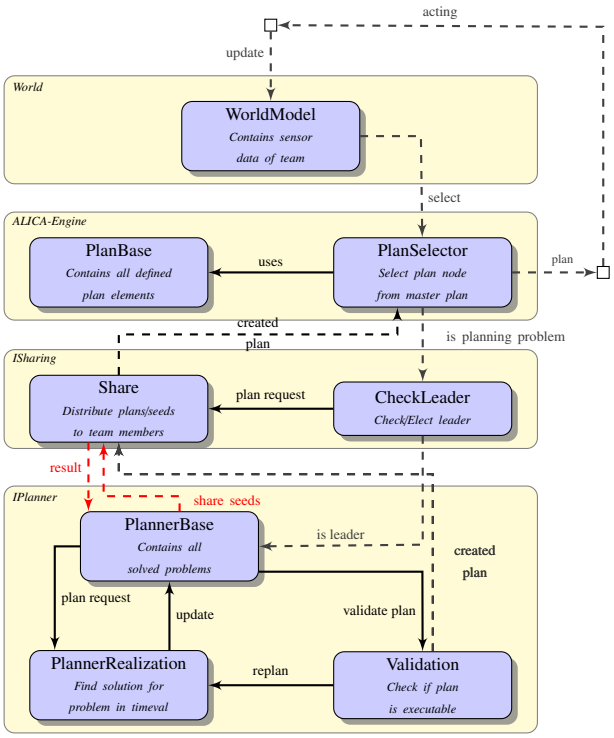

Figure 3. pRoPhEt MAS

5 Evaluation

We used NASA's Mars Rover domain to evaluate MAGiC. This problem is used as a benchmark from the International Planning Competition (IPC-2002). In this problem, rovers have to collect soil, rock samples, and communicating back to the lander. This problem is quite similar to the IMPERA [10] domain. A team of robots has to explore the environment to find stones and carry them to certain location for building specific structures. In IMPERA, we use the language ALICA [11] for the description and execution of multi-agent plans. It is a language for specifying multi-agent plans together with operational semantics. It provides modeling facilities for cooperative behaviors. Furthermore, the language can be used in highly dynamic domains [9] and can estimate their teammates' decisions, act upon them, and correct them when new information is available. ALICA is developed, expanded, and evaluate in dynamic robotic applications like RoboCup[3] since 2008 [11].

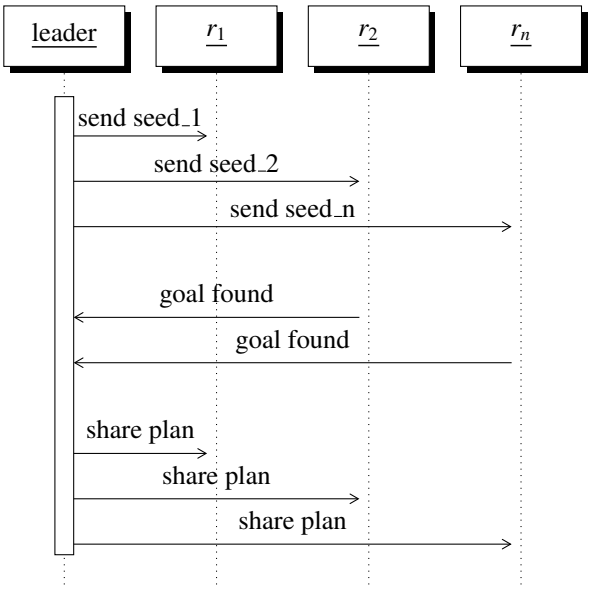

Figure 4. Share seeds to accelerate search

The evaluation of the IPC problems is shown in Table 1. For all planning problems we ran a modified Fast Downward Planner (FDP) [2], which divides the search space to the amount of agents in team. Moreover, we add the results of Multi-Agent Forward Search (MAFS) [1] into the table. All lines with minus were not evaluated by MAFS. Both using eager best-first search and an alternation open list with one queue for each of the two heuristic functions *ff* [12] and *context-enhanced additive heuristic* [1]. Experiments were run on an Intel i7-2630QM CPU 2.00GHz processor, the time limit was set to 30 minutes, and memory usage was limited to 4GB. For all problems Fast Downward Planner is 24.7% faster than MAFS on average, but MAFS got in average 9.2% better resulting costs.

For communication the leader needs to distribute the problem to all n agents. These agents return the result. Finally, the central agent distributes the final solution. The upper communication bound is for n agents is limited to $3*(n-1)$. In average FDP requires 98.2% lesser messages than MAFS for the evaluated problems "Rovers5" to "Rovers17".

We like to analyze the improvement solving the problem by single agent compare to cooperative problem solving. In figure 5 we compare the calculation time of a single robot versus multi robot for FDP. The figure contains as well the maximum calculation time of the team. We can see that MAGiC with FDP performs on average 46.4% better than with single agent. In worst case single

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

Table 1. : Table contains an evalutation for the NASA's Mars Rover domain of Multi-Agent Forward Search (MAFS) and Fast Downward Planner (FDP). Solution cost (steps); running time (in sec.) and the number of sent messages are shown.

problem	**# agents**	**Costs**		**Runtime**		**Messages**	
		MAFS	FDP	MAFS	FDP	MAFS	FDP
Rovers1	1	-	10	-	0.046	-	0
Rovers2	1	-	8	-	0.037	-	0
Rovers3	2	-	13	-	0.05	-	3
Rovers4	2	-	10	-	0.054	-	3
Rovers5	2	**22**	**22**	0.13	**0.069**	84	**3**
Rovers6	2	**37**	38	**0.07**	0.074	27	**3**
Rovers7	3	**18**	**18**	**0.07**	0.076	225	**6**
Rovers8	4	**26**	28	0.2	**0.143**	937	**6**
Rovers9	4	38	**34**	0.82	**0.206**	380	**6**
Rovers10	4	38	**36**	0.41	**0.251**	271	**6**
Rovers11	4	**37**	38	**0.34**	0.382	299	**6**
Rovers12	4	**21**	22	0.09	**0.05**	435	**6**
Rovers13	4	49	**46**	**0.15**	0.272	472	**6**
Rovers14	4	**31**	32	0.42	**0.208**	310	**6**
Rovers15	4	46	**44**	0.33	**0.306**	252	**6**
Rovers16	4	-	43	-	0.374	-	6
Rovers17	6	**52**	**52**	0.57	**0.3**	628	**15**
Rovers18	6	-	48	-	1.36	-	15
Rovers19	6	-	74	-	16.099	-	15
Rovers20	8	-	87	-	9.673	-	21
Rovers21	6	-	68	-	5.887	-	15
Rovers22	6	-	69	-	15.93	-	15
Rovers23	8	-	80	-	16.493	-	21
Rovers24	8	-	122	-	214.782	-	21
Rovers25	10	-	31	-	2.418	-	27
Rovers26	10	-	62	-	5.682	-	27
Rovers27	10	-	103	-	73.524	-	27
Rovers28	10	-	78	-	28.361	-	27
Rovers29	10	-	60	-	22.616	-	27
Rovers30	10	-	115	-	113.091	-	27
Rovers31	12	-	140	-	1168.313	-	33
Rovers32	12	-	0	-	0	-	33

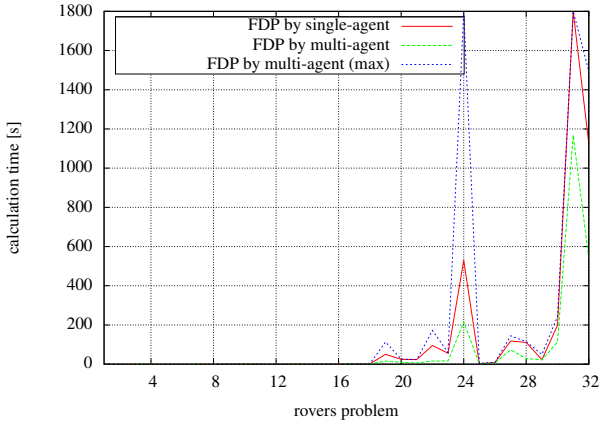

Figure 5. Presents calculation time of modified Fast Downward Planner for rovers' domain

agent solving performs on average 45% better, which can happen if nearly every robot breaks down. In problem 24 some robots does not found a solution. In problem 31 the single agent was not able to solve the problem. The search was interrupted after 30 minutes. Finally, the cooperative solving was successful.

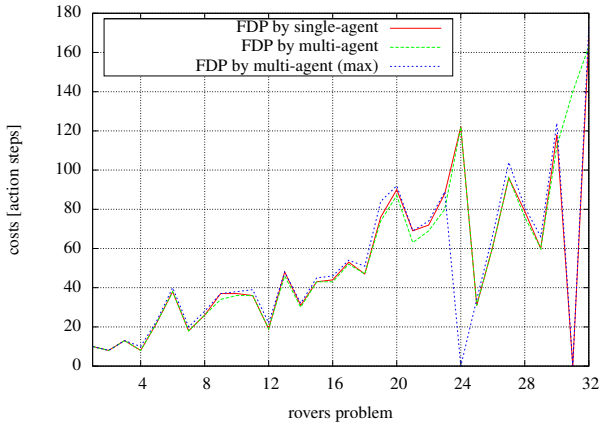

Figure 6. Illustrate costs of modified Fast Downward Planner for rovers' domain

In figure 6 we compare the cost of a single robot with multi robot. Furthermore, the figure contains the maximum costs of the team. We can see that MAGiC with FDP performs on average 2.8% better than single agent solving. In worst case MAFS performs 4.9% better.

6 Conclusions

Using autonomous mobile robots creates new potential for applications, if autonomous mobile robots divide the planning process to all team members. Hence, it creates the opportunity to optimize scenarios like monitoring, disaster management, logistics operations, extraterrestrial missions 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 problems the planning is not feasible because of memory limitation. Thus, distributed planning is an important part of multi-agent systems.

We use all agents as planning resources to reduce the planning time and divide the memory usage. If the team has to determine a plan, the team leader will start the search for solution and share seeds with all team members. If a member of the team finds a solution, he shares it with the team.

In our evaluation we took the rovers' scenario of the IPC2002 to compare our planning system to the state of the art planner. We were able to improve the search time by 46.4% and the costs by 2.8% on average in the IPC rovers' problems and we were able to reduce the messages by 98.2% compare to MAFS.

Our next steps are to evaluate the framework up to high numbers of teammates. Furthermore, another goal is to evaluate a free kick with up to 10 agents in the RocoCup MSL[4] domain.

7 Acknowledgments

We thank all members of the IMPERA project at the University of Kassel for their contributions.

Project IMPERA is funded by the German Space Agency (DLR, Grant number: 50RA1112) with federal funds of the Federal Ministry of Economics and Technology (BMWi) in accordance with the parliamentary resolution of the German Parliament.

References

[1] R. Nissim and R. I. Brafman., "Distributed heuristic forward search for multi-agent systems", CoRR abs/1306.5858, 2013.

[2] M. Helmert., "The Fast Downward Planning System", Journal of Artificial Intelligence Research 26, 2006.

[3] Anderson, David P., "BOINC: A System for Public-Resource Computing and Storage", Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, pp. 4-10, 2004.

[4] http://www.robocup.org/robocup-soccer/middle-size/

[4] Michael Brenner and Bernhard Nebel., “Continual planning and acting in dynamic multiagent environments”, Autonomous Agents and Multi-Agent Systems, 19(3):297–331, 2009.

[5] Daniel Saur and Kurt Geihs, “pRoPhEt MAS: Reactive planning engine for multi-agent systems”, 13th International Conference on Intelligent Autonomous Systems, 2014.

[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] G. Antonelli, F. Arrichiello, and S. Chiaverini., “Experiments of formation control with collisions avoidance using the null-space-based behavioral control.”, Mediterranean Conference on Control and Automation, 0:1–6, 2006.

[8] E. Tuci, R. Groß, V. Trianni, F. Mondada, M. Bonani, and M. Dorigo., “Cooperation through self-assembly in multi-robot systems”, ACM Trans. Auton. Adapt. Syst., 1(2):115–150, 2006.

[9] H. Skubch, M. Wagner, R. Reichle, and K. Geihs., “A modelling language for cooperative plans in highly dynamic domains”,Mechatronics, 21:423–433, 2011.

[10] Eich, M. et al. “Towards Coordinated Multi-Robot Missions for Lunar Sample Collection in Unknown Environment”, J. Field Robotics, Journal of Field Robotics, 2013, pp. 1556-4967.

[11] Skubch, H., “Modelling and Controlling of Behaviour for Autonomous Mobile Robots”, Westdeutscher Verlag GmbH, 2013.

[12] Hoffmann, J. and Nebel, B.,The FF Planning System: Fast Plan Generation Through Heuristic Search, J. Artif. Int. Res., AI Access Foundation, USA, 2001, pp. 253-302.

[13] Skubch, H., Wagner, M., Reichle, R., Triller, S., and Geihs, K.: Towards a comprehensive teamwork model for highly dynamic domains. In Filipe, J., Fred, A., and Sharp, B., editors, *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence*, volume 2, page 121–127. INSTICC Press, INSTICC Press (2010)