MAPmAKER: Performing Multi-Robot LTL Planning Under Uncertainty

Sergio García*, Claudio Menghi†, and Patrizio Pelliccione*‡,
*Chalmers | University of Gothenburg, Gothenburg (Sweden)
† University of Luxembourg, Luxembourg City (Luxembourg)
‡ University of L'Aquila, L'Aquila (Italy)
Email: sergio.garcia@gu.se, claudio.menghi@uni.lu, patrizio.pelliccione@gu.se

***Abstract*—** Sergio ►*rewrite*◄ **Robot applications are being increasingly used in real life to help humans performing dangerous, heavy, and/or monotonous tasks. They usually rely on planners that given a robot or a team of robots compute plans that specify how the robot(s) can fulfill their missions. Current robot applications ask for planners that make automated planning *tractable* and possible even when only *partial knowledge* about the robot application is present, e.g., some information about the environment in which the robots are deployed is missing.**

This paper presents MAPmAKER, a tool that aims to support run-time mission execution by tackling the previous challenges, i.e., it provides a decentralized planning solution that helps to reduce the planning overhead and is able to work when only partial knowledge of the environment is present. Decentralization is realized by decomposing the robotic team into subclasses based on their missions, and then by running a classical planning algorithm. Partial knowledge is handled by calling several times a classical planning algorithm.

Demo video available at: https://youtu.be/TJzC_u2yfzQ

I. Introduction

Robotic applications usually rely on a set of robots that are used to perform missions. The term mission can refer to a *global mission*, i.e., the high-level mission that must be accomplished by the whole team [1] or a *local mission*, i.e., the mission that should be achieved by a single robot, possibly by collaborating with other robots [2]. Planners are one of the main ingredients that allow robots to achieve missions. A *planner* is a software component that receives as input a model of the robotic application and computes a set of actions—a *plan*—that, if performed, allows the achievement of a desired mission [3].

Current robotic applications require planners to address two main challenges: 1) the planning problem should be solved by using algorithms that make the problem tractable; 2) the planning algorithm should work also when (only) partial knowledge about the system—the robots and their working environment—is present.

Sergio ►*in the following paragraphs I cite several works to speak trying to give some background but also related work. Is it enough?*◄ Tractability refers to the capability of computer algorithms in solving problems. Several works studied centralized planners that are able to manage *teams* of robots that collaborate to achieve a certain goal (a global mission) [1], [4], [5]. However, planning is computationally expensive, especially when the number of robots within the team is increased and they need to collaborate to fulfill their local missions. For this reason, research interest had focused on decomposing a global mission into a set of local missions to be achieved by each robot of the team [2], [6], [7]. These local missions have been recently exploited by *decentralized* planners [2], i.e., planners that instead of evaluating the global mission over the whole team of robots, analyze the satisfaction of local missions inside a subset of the team of robots. In this way, the problem of finding a collective team behavior is decomposed into sub-problems that avoid the expensive fully centralized planning. However, the applicability of these algorithms has never been studied when only partial knowledge about the system is available.

Sergio ►*planners with ltl*◄
The role of partial knowledge in software development has been strongly studied in literature. Research has been done on how to consider partial knowledge in requirement analysis and elicitation [8]–[10], in the development of a model of the system that satisfies a set of desired properties [11]–[15] and in checking whether an already designed model possesses some properties of interest [16]–[18]. However, most of the existing planners assume that the environment in which the robots are deployed is known [19]. This assumption does not usually hold in real-word scenarios [20], where, for example, the robots navigate in environments affected by natural disasters, where the movement between locations or the execution of specific actions may be impossible due to structural collapses, flooding, etc. The planners that consider partial information about the environment in which the robots operate (e.g., [21]–[23]) usually rely on probabilistic algorithms and are not *decentralized*.

This work presents MAPmAKER: a Multi-robot plAnner for PArtially Known EnviRonments. MAPmAKER provides a *decentralized* planning solution that works in *partially known* environments. Decentralization is realized by decomposing the robotic team into subteams based on their missions, and then by running a classical planning algorithm. Partial knowledge is handled by calling twice a classical planning algorithm. The theory that supports MAPmAKER including proofs of correctness, a detailed description of the modelling formalisms and the verification procedures can be found in [24]. We developed MAPmAKERas proof of concepts to show how planners can deal with the previously stated features rather than as a tool to be used for real-world scenarios. MAPmAKER builds upon the planner proposed by Tumova et al. [2]. **Sergio** ►*come back to this sentence after checking evaluation*◄ MAPmAKER is evaluated by analysing its behaviour on a robot application obtained from the RobotCup Logistics League competition [25] and on a robotic application working in an apartment of about 80 m^2 [26]. MAPmAKER together with 1) a complete replication package, 2) a set of videos showing MAPmAKER in action computing and solving the scenarios presented at the previous bullet, and 3) a brief user guide that defines the functionalities provided by our tool are available at https://github.com/claudiomenghi/MAPmAKER/.

This paper is organized as follows. Section II presents an overview of MAPmAKER. Section III describes how MAPmAKER can be used. Section IV evaluates MAPmAKER. Section V concludes with final remarks.

II. MAPMAKER'S OVERVIEW

An overview of MAPmAKER is depicted in Fig. 1. MAPmAKER's planner takes as input the models of the robots (1) and of the environment in which they are deployed (2) and the mission each robot should achieve (3). Both the models of the robots and their environment may be partial since there can be uncertainty about information contained in these models. MAPmAKER uses the model of the environment and the robots to compute plans that allow the achievement of missions using an appropriate planner. The implemented planner is able to compute plans that definitely ensure the mission satisfaction, i.e., definitive plans (4), and plans that may ensure property satisfaction since they depend on some partial knowledge present in the models of the robots and the environment (5). More precisely, a *definitive plan* is a sequence of actions—e.g., move from *a* to *b*—that ensure the satisfaction of the local mission for each robot. A *possible plan* is a sequence of actions that may satisfy the local mission due to some unknown information about the model of the robots or the environment in which they are deployed. If MAPmAKER is not able to find neither a definitive nor a possible plan a message is sent to the user (6). Otherwise, an appropriate component is used to choose between definitive and possible plans (if both are present) or simply chooses the possible plan if no definitive plan is present. Definitive plans are not present when the only way to satisfy the local mission is based on some unknown information about the model of the robots or the environment in which they are deployed. MAPmAKER then executes the selected plan (7).

As robots perform plans, information about uncertain parts of the model is detected. MAPmAKER updates the models with the detected information (8) and if it detects that a plan is not anymore executable, the planner is re-executed (9).

In the following, we provide some additional information about the inputs processed by MAPmAKER, the planning algorithm, the selection between definitive and possible plans and how models are updated when information about uncertain parts is detected.

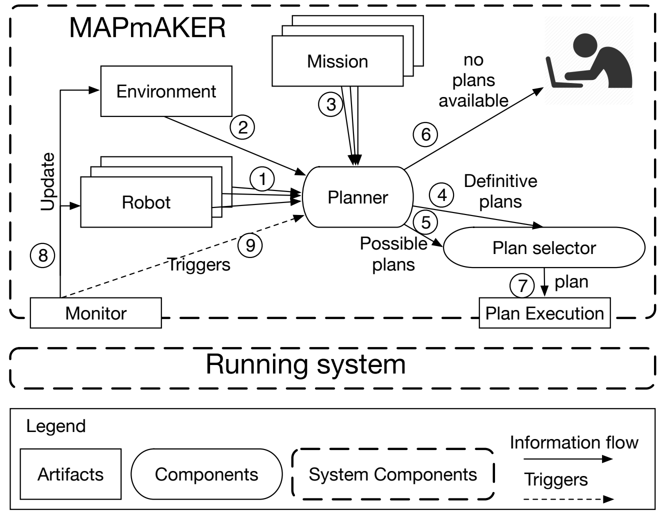

Fig. 1. Overview of MAPmAKER.

Models of the robots and their environment. The models of the robots and their environments are provided using a specific form of transition system that allows the specification of uncertain parts; further information might be found in [24]. These models describe the initial positions of the robots, the map describing the environment where robots are deployed, and how robots can move between different locations of the map. Furthermore, the proposed models embed partial knowledge as follows:

• *Partial knowledge about the actions execution.* The execution of certain actions is uncertain, meaning that it is unclear whether an action can be executed. This type of partial knowledge allows specifying that the transition between two of the cells that conform the grid map of the environment (see Fig. 2) can be: always possible, always impossible (i.e. a wall), not known (i.e. a door between two rooms that can be open or closed).

• *Unknown service provisioning.* It is unclear whether a service—i.e., "events of interest associated with execution of certain actions rather than over atomic propositions" [7]—can be provided or not in a specific location. For example, it is unclear whether a robot can take a picture of an item in a given map location. This uncertainty may be caused for example by the presence of an unexpected object that covers the robot visual in that location.

• *Unknown meeting capabilities.* Robots can meet and synchronize in certain locations. For example, it is unclear whether two robots can exchange a load in a given map location. This uncertainty may be caused by a collapsing registered in the environment where the robots are deployed.

Mission specification. Each robot is able to perform a complex mission, which is specified using an LTL formula. This formula specifies how the services must be provided by the robots. For example, a mission for a robot r_1 may require r_1 to periodically load debris on r_2. Thus, in order to allow robot r_1 to fulfill its mission, it is necessary that robots r_1 and r_2 synchronize their behaviours.

Planning. The *Planner* uses the models of the robot(s) and the environment in order to compute plans that allow satisfying the missions of the robots. The planner distributes the robots within the robotic application into subteams —that we call "dependency classes"— based on the mission that each robot has to achieve. Each dependency class contains a subset of robots that depend on each other for achieving their missions. After dependency classes are computed they are considered in isolation regarding the computation of plans that allow robots to satisfy their missions.

To compute a plan for a dependency class the LTL formulae that are used to describe missions are evaluated on partial models. Possible and definitive plans are computed by executing a classical planning algorithm twice: once for computing possible plans and once for computing definitive plans.

Choosing between definitive and possible plans. The plan selector component aims at choosing between possible and definitive plans. Several policies can be applied to choose between these plans. Possible plans can be chosen only in cases in which a definitive plan is not present. Another policy may choose the plan with the shortest length, or it may consider non-functional aspects of the plans e.g., time to perform certain actions, or likelihood of detecting true or false evidence about partial information. In this work we assume that the planner always chooses the shortest between the possible and the definitive plan. This policy may, for example, reduce energy consumption, since every action performed by the robots may consume energy.

Detection of uncertain information. As robots perform actions and navigate within the environment, information regarding uncertain services and meeting capabilities can be detected. Specifically, robots detect whether actions, services, and meeting capabilities are executable, provided, and possible, respectively. MAPmAKER updates the models of the robots and of the environment with the detected information. Then, if needed, the planning algorithm is triggered and re-executed.

III. MAPMAKER IN ACTION

MAPmAKER was developed as a MATLAB [27] standalone application. It is developed on top of an existing planner —presented in [2]— which has been chosen since it already implements a decentralized planning procedure. MAPmAKER calls this planner twice considering two different versions of the model of the robots and their environments. The results obtained by performing this procedure are sound and correct. Additional details and proofs can be found in [24].

MAPmAKER can be executed in two ways, as shown in Listing 1. The first option is used to compute plans for custom missions in custom models of environment and robots. `robots` is a variable that specifies the number of robots and their models. Then, `environment` is a variable that contains a model of the environment and its uncertainty. Finally, `missions` contains the local mission to be achieved by each robot. With the second option, we provide a way of replicating the experiments presented in this paper and in [24]. `Scenario` is a Matlab file containing the model of the environment and the robots and `Experiment` encodes the mission that must be satisfied by each robot. Then, the research question `RQ` with which the two previous variables are associated must be specified —e.g. RQ1 or RQ2.

```
mapmakerRunner(robots, environment , missions);
mapmaker_exp('Scenario', 'Experiment', 'RQ')
```

Listing 1. Running MAPmAKER

When MAPmAKER is executed a graphical interface similar to the one presented in Figure 2 is showed. The figure is a screenshot showing the performance of our tool where we changed the size of some numbers and added the plans for helping the reader. The grid represents the environment in which the robots are moving. Each cell represents a location of the environment and has a number associated, as labeled in some of them. Robots are represented by squared colored boxes. Actions are used to encode movements, i.e., each robot can move left, right, up, and down. A robot cannot move between adjacent cells if they are separated by thick bordered lines. Whenever it is unclear if a robot can move between adjacent cells, these cells are separated by a red border. Whenever a service can be provided by a robot in a cell, the cell is labeled with the associated number and the color of the corresponding robot. Finally, synchronization capabilities are represented by a black cross. If it is unclear if two robots can synchronize in a cell, the cell is labeled with a green cross. The graphical interface shows the plan execution. Assume for example that the robots r_1, r_2, and r_3 have the following *local missions*. Robot r_1 has to perform service 1, which is provided in cells 7 and 9. Service 1 and service 2 (which has to be accomplished by robot r_2) are located in a cell labeled with a cross, so robots must meet there and perform both actions at the same time. Robot r_1 can also perform this mission in cell 9, but the synchronization in this cell is unsure. Robot r_2 must synchronize with robot r_1 and perform service 2, but then it has to reach cell 22 for performing action 3. Finally, robot r_3 has to perform service 4 in cell 2 and service 5 in cell 18 or in cell 30.

In Figure 2 we show different plans that can be performed by the robots. They all accomplish a number of actions in a periodic fashion. Robot r_1 could accomplish two different plans. P1 represents a definitive plan and P1' a possible plan where a true evidence was detected by the robot. In both paths, the robot reaches the cell where the requested service can be performed with the help of robot r_2. For robot r_2 we show two different paths as well. P2 represents a possible plan —due to the uncertainty of the transition between cells 14 and 20— where r_2 must synchronize with r_1 in order to accomplish service 2. P2' is a similar path with the difference that the synchronization is not assured in the associated cell. Finally, P3 represents a possible path where robot r_3 accomplishes service 4 and service 5 in cell 18.

The following missions were considered: 1) robot r_1 must achieve the mission $\mathsf{F}(s_1 \wedge (\mathsf{F}(s_2 \vee s_3)))$. It had to reach a predefined destination where service s_1 is provided, and then perform either service s_2 or service s_3. 2) robot r_2 must achieve the following mission $\mathsf{G}(\mathsf{F}(s_4 \vee s_5))$. Furthermore, it aims at

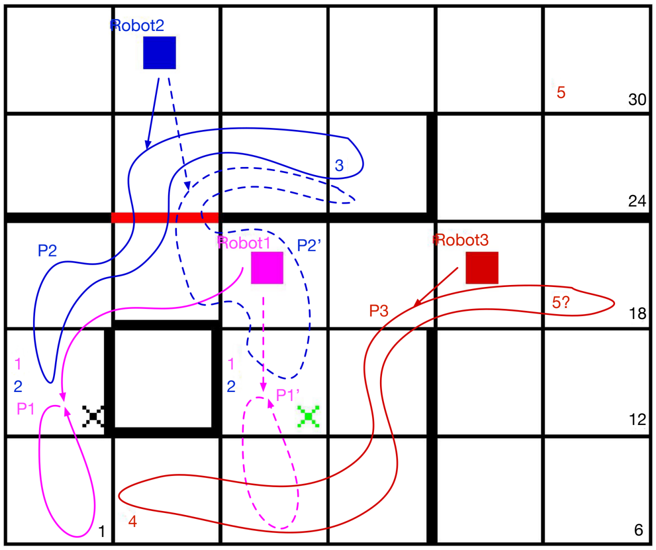

Fig. 2. MAPmAKER usage scenario.

helping r_1 in providing service s_1, i.e., robots r_1 and r_2 must meet in cells where service s_1 is provided.

IV. Evaluation

To evaluate MAPmAKER we considered the following research questions: **RQ1**: How does MAPmAKER help planning in partially known environments? **RQ2**: How does the employed decentralized algorithm help in plan computation?

To answer RQ1 we had considered a set of existing examples: one obtained from the RoboCup Logistics League competition [25] and an apartment of a large residential facility for senior citizens [26]. We created a partial robot application starting from the models of the robots and their environment contained in these examples. We performed different experiments in which we evaluated the impact of partial information about the action execution, services provisioning and meeting capabilities on the planning procedure. We compare whether computing possible plans actually helps mission achievement. This is done by comparing our planner with one that is only able to compute definitive plans. The results can be summarized as follows. MAPmAKER is effective when a possible plan is selected, and the robot discovers during the plan execution that unknown actions, services, and meeting capabilities are executable, provided, and possible, respectively. MAPmAKER is also effective when this three conditions are given: 1) a possible plan is computed; 2) the possible plan can actually be performed; and 3) a classical planning algorithm cannot compute a definitive plan. This situation occurs in the cases in which the only way to fulfill a mission involves some partial information present in the model. When MAPmAKER chooses a definitive plan, it is as effective as a classical planner that is only able to compute definitive plans. The computation time and the length of plans are increased whenever a possible plan is chosen but unknown actions, services, and meeting capabilities turned to be not executable, provided, and possible, respectively. More information about the considered examples, experimental set up, and the obtained results can be found in [24].

To answer RQ2 we analyzed the advantages of the decentralized procedure provided by MAPmAKER. We had considered the set of partial models considered in the previous experiments. We added an additional robot, i.e., robot r_3, which has a mission that can be achieved without collaborating with neither robot $r1$ nor with robot $r2$. We then executed MAPmAKER with the decentralized procedure enabled and disabled. Then, we compare each performance. When MAPmAKER was executed with the decentralized procedure it computed two dependency classes; one containing robots r_1 and r_2 and one containing robot r_3. Viceversa, when the decentralized procedure was disabled, MAPmAKER analyzed a single team containing robots r_1, r_2, and r_3. The results show a drastic improvement in the efficiency of MAPmAKER when the decentralized procedure was enabled. More information can be found in [24].

V. Conclusions

We presented MAPmAKER, a decentralized planner for partially known environments. The theoretical results show that any planner can be used within MAPmAKER. It is realized as a proof of concepts to show that (i) models containing partial information can be efficiently handled in by current planners and that (ii) decentralized procedures help in improving performances. The current implementation relies on a naive implementation of a planner that comes from literature and has been customized within the proposing framework. Our evaluation showed how MAPmAKER improves planning in cases in which partial information is present. We also showed that the implemented decentralized procedure improves the performance of the planning algorithm.

References

[1] M. M. Quottrup, T. Bak, and R. Zamanabadi, "Multi-robot planning: A timed automata approach," in *ICRA*, vol. 5. IEEE, 2004.

[2] J. Tumova and D. V. Dimarogonas, "Multi-agent planning under local ltl specifications and event-based synchronization," *Automatica*, 2016.

[3] J.-C. Latombe, *Robot motion planning*. Springer, 2012, vol. 124.

[4] M. Kloetzer, X. C. Ding, and C. Belta, "Multi-robot deployment from ltl specifications with reduced communication," in *Conference on Decision and Control and European Control Conference*. IEEE, 2011.

[5] S. G. Loizou and K. J. Kyriakopoulos, "Automated planning of motion tasks for multi-robot systems," in *CDC*. IEEE, 2005.

[6] P. Schillinger, M. Bürger, and D. Dimarogonas, "Decomposition of finite ltl specifications for efficient multi-agent planning," in *International Symposium on Distributed Autonomous Robotic Systems*, 2016.

[7] M. Guo and D. Dimarogonas, "Multi-agent plan reconfiguration under local ltl specifications," *The International Journal of Robotics Research*, 2015.

[8] C. Menghi, P. Spoletini, and C. Ghezzi, "Integrating goal model analysis with iterative design," in *REFSQ*. Springer, 2017.

[9] ——, "Cover: Change-based goal verifier and reasoner," in *REFSQ Workshops*. Springer, 2017.

[10] E. Letier, J. Kramer, J. Magee, and S. Uchitel, "Deriving event-based transition systems from goal-oriented requirements models," *Automated Software Engineering*, vol. 15, no. 2, 2008.

[11] S. Uchitel, G. Brunet, and M. Chechik, "Synthesis of partial behavior models from properties and scenarios," *TSE*, vol. 35, no. 3, 2009.

[12] S. Uchitel, D. Alrajeh, S. Ben-David, V. Braberman, M. Chechik, G. De Caso, N. D'Ippolito, D. Fischbein, D. Garbervetsky, J. Kramer, *et al.*, "Supporting incremental behaviour model elaboration," *Computer Science-Research and Development*, vol. 28, no. 4, 2013.

[13] M. Famelis, R. Salay, and M. Chechik, "Partial models: Towards modeling and reasoning with uncertainty," in *ICSE*, 2012.

[14] A. Albarghouthi, A. Gurfinkel, and M. Chechik, "From under-approximations to over-approximations and back," in *TACAS*, 2012.

[15] A. Bernasconi, C. Menghi, P. Spoletini, L. D. Zuck, and C. Ghezzi, "From model checking to a temporal proof for partial models," in *Software Engineering and Formal Methods*. Springer, 2017.

[16] C. Menghi, P. Spoletini, and C. Ghezzi, "Dealing with incompleteness in automata-based model checking," in *Formal Methods*, 2016.

[17] G. Bruns and P. Godefroid, "Model checking partial state spaces with 3-valued temporal logics," in *CAV*, 1999.

[18] M. Chechik, B. Devereux, S. Easterbrook, and A. Gurfinkel, "Multi-valued symbolic model-checking," *ACM TOSEM*, 2004.

[19] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson, "Mpdm: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving," in *ICRA*, 2015.

[20] M. Lahijanian, M. Maly, D. Fried, L. Kavraki, H. Kress-Gazit, and M. Vardi, "Iterative temporal planning in uncertain environments with partial satisfaction guarantees," *IEEE Transactions on Robotics*, 2016.

[21] N. Roy, G. Gordon, and S. Thrun, "Planning under uncertainty for reliable health care robotics," in *Field and Service Robotics*. Springer, 2006.

[22] N. E. Du Toit and J. W. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Transactions on Robotics*, 2012.

[23] J. F. Diaz, A. Stoytchev, and R. C. Arkin, "Exploring unknown structured environments." in *FLAIRS Conference*. AAAI Press, 2001.

[24] C. Menghi, S. Garcia, P. Pelliccione, and J. Tumova, "Multi-robot ltl planning under uncertainty," in *International Symposium on Formal Methods*. Springer, 2018, pp. 399–417.

[25] C. Karras, U. Deppe, T. Neumann, T. Rohr, A. Niemueller, W. Uemura, D. Ewert, N. Harder, S. Jentzsch, N. Meier, and S. Reuter, "Robocup logistics league rules and regulations," 2016.

[26] 2014. [Online]. Available: http://angeninnovation.se

[27] MATLAB, https://mathworks.com/products/matlab.html, 2017.