

# MAPmAKER: A Tool for Performing Multi-Robot LTL Planning Under Uncertainty

Anonymous Author(s)

## ABSTRACT

Robot applications are increasingly asking for decentralized techniques that allow for tractable automated planning. Furthermore, those applications can be deployed in dynamic environments where its uncertainty must be handled. Typically, environments where human beings are involved can just provide a partial knowledge of its model, i.e. the current state of a door between two rooms in uncertain.

Our proposed tool, MAPmAKER tackle the limitations that current planning techniques are used for teams of robots: (1) it decomposes the robotic team into subclasses, avoiding the not scalable centralized approach; (2) it considers complex-high level missions given in temporal logic; (3) it is able to work also with only partial knowledge of the environment, performing possible plans.

### ACM Reference format:

Anonymous Author(s). 2018. MAPmAKER: A Tool for Performing Multi-Robot LTL Planning Under Uncertainty. In *Proceedings of 40th International Conference on Software Engineering, Gothenburg, Sweden, May 27–June 3, 2018 (ICSE 2018)*, 4 pages. DOI: 10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

MAPmAKER provides a planner where a robot application is defined using finite transition systems. 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 [17]. Each robot application contains the robots that conform the team and the mission that they have to achieve.

A *global mission* represents the high-level mission that must be accomplished by the whole team [13, 20, 26] and that is decomposed into a set of *local missions* [11, 11, 29, 31]. Every robot is commanded to achieve a local mission, specified as a LTL property. As seen in [31], this collaborative fashion of accomplishing the global mission is performed in a *decentralized* way. Each robot that is part of a subset of the team computes the solution for its own sub-mission, avoiding the expensive fully centralized planning and making it more robust to local problems.

Nowadays, most of the planners consider the model of the environment as known and not dynamic [6]. However, this is not a real condition of real world scenarios, where only *partial knowledge* can be ensured. For this reason, our tool is able to compute a plan even

when only partial information of the environment is available, as seen in [7, 10, 27]. However, the novelty of our work consists in fuse all this features, exploiting a *decentralized* methodology. This kind of approaches are not yet studied in detail, due to there are only a few planners managing this issues [11].

**Organization.** Section 2 introduces robotic applications by highlighting the status of current planners. Section 3 describes the MAPmAKER approach. Section 4 presents the MAPmAKER tool. Section 5 concludes with final remarks.

## 2 LIMITATIONS OF CURRENT PLANNERS

### Decentralized solutions.

**Sergio** ▶ Not sure if we should add some planner tools as V-Rep <http://www.coppeliarobotics.com>. ROS Moveit <http://moveit.ros.org/>, Gazebo <http://gazebo.org/> or look for papers presenting tools◀

Decentralized planning problem has been studied for known environments [11, 29, 31]. However, planners for partially known environments do not usually employ decentralized solutions [7, 10, 27].

*Dealing with partial knowledge in planning.* Planning in partially known environments is handled in different ways. (1) Several works (e.g., [2, 3, 5, 8, 10, 15, 22, 24, 28, 32]) consider probabilities within the planning algorithm. Most of these works treat partial information by modeling the robotic application using some form of *Markov decision processes* (MDP). In some of these works [5, 8] transitions of the robots are associated with probabilities which indicate the probability of reaching the destination of the transition given that an action is performed. In other works [32], transition probabilities are not exactly known but are known to belong to a given uncertainty sets. Finally, several works [15, 28] consider partially observable Markov decision processes. All these approaches generally generate plans that maximize the worst-case probability of satisfying a mission. Differently, our work does not consider probabilities. (2) Several works (e.g., [6, 16, 18, 19, 23]) studied how to change the planned trajectories when unknown obstacles are detected or when obstacles move in a unpredictable way. In this case, the used underlying model is some sort of *hybrid model*, i.e., models in which finite state machines are combined with differential equations. In [16], to plan trajectories the authors use a high-level planner that exploits an abstraction of the hybrid system and the mission to compute high-level plans. The low-level planner uses the dynamics of the hybrid system and the suggested high-level plans to explore the state space for feasible solutions. Every time an unknown obstacle is encountered, the high-level planner modifies the coarse high-level plan online by accounting for the geometry of the discovered obstacle. Within this framework, MAPmAKER can be considered as a high-level planner that is able to use an abstraction of the hybrid system that contains partial information, i.e., encode unknown obstacles. (3) Some approaches analyzed how to update plans when

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

ICSE 2018, Gothenburg, Sweden

© 2018 ACM. 978-x-xxxx-xxxx-x/YY/MM. ...\$15.00

DOI: 10.1145/nnnnnnn.nnnnnnn

new information about known model of a robotic application is detected (e.g., [11]). Differently, in our approach portions of the model of the robotic application are partially known, partial knowledge is reduced as true and false evidence about partial information is detected. Other works (e.g., [1]), aim at detecting how to explore totally unknown environments. (4) Plan synthesis is a particular instance of controller synthesis. Controller synthesis (e.g., [4, 9]) aims at finding a component, usually indicated as controller or supervisor, that ensures property satisfaction for all the possible system executions. Differently, plan synthesis aims at finding a single execution, i.e., a plan that ensures property satisfaction. The controller synthesis is usually ([5, 12, 14, 19, 33]) performed by solving a two player game between robots and their environment. The goal is to find a strategy the robots can use that allows always winning the game. Differently, in our case the planning algorithm ensures that there is a way of completing the *single* (possible) plan that satisfies the property of interest. (5) MAPmAKER can be classified on the boundary between reactive synthesis [5, 19, 30] techniques and iterative planning [12, 21]. As reactive synthesis techniques, MAPmAKER constructs a control strategy that accounts for every possible variation in the environment, but the computed plan does not allow always winning the *two player game* between the robots and their environment. As iterative planning, a new plan is computed on-the-fly when new information is available.

### 3 THE MAPMAKER APPROACH

–Explain the tool from a high-level point of view–

This work presents MAPmAKER (Multi-robot pLanner for Partially Known EnviRonments), a *novel decentralized* planner for partially known environments. Given a team of robots and a local mission for each robot, MAPmAKER partitions the set of robots into classes based on dependencies dictated by the local missions of each robot. For each of these classes, it explores the state space of the environment and the models of the robot searching for definitive and possible plans. A *definitive plan* is a sequence of actions 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. MAPmAKER chooses the plan that allows the achievement of the mission by performing the lower number of actions, but other policies can also be used.

**Specific contributions.** Specific contributions are detailed in the following: (1) we define the concept of *partial robot model*, which allows the description of the behavior of the robots and its environment when only partial information is available. Specifically, a partial robot model allows considering three types of partial information: partial knowledge about the execution of transitions (possibility of changing the robot location), on service provision (whether the execution of an action succeed in providing a service) and on the meeting capabilities (whether a robot can meet with another); (2) we define the concept of local mission satisfaction for partial robot models; (3) we define the distributed planning problem for partially specified robots; (4) we propose a distributed planning algorithm and we proved its correctness; The results show the effectiveness of the proposed algorithm.

The presence of partial knowledge about the robots and their environment is described in the following.

**Partial knowledge about the actions execution.** The robots can move between cells separated by grey lines, while they cannot cross black bold lines. It is unknown whether it is possible to move between cells  $c_{14}$  and  $c_{20}$ . This is indicated using a dashed black bold line. It is also unknown whether robot  $r_3$  can send pictures using a communication network in location  $l_3$  and specifically in cell  $c_{18}$ , i.e., whether action  $s_p$  can be performed. Locations of the environment where it is unknown if an action can be provided are marked with the name of the action preceded by symbol ?.

**Unknown service provisioning.** There are cases in which actions can be executed but there is uncertainty about service provisions. For example, actions  $ud1$  and  $ud2$  of robot  $r_2$  unload the robot. Action  $ud2$  will always be able to provide the *unload* service, while it is unknown whether  $ud1$  is actually able to provide this service since its effectiveness depends on the size of the collected debris. In Fig. 2, the label  $L(ud1, unload) = ?$  indicates that there is partial knowledge about the provision of the *unload* service when action  $ud1$  is performed.

**Unknown meeting capabilities.** It is unknown whether robots  $r_1$  and  $r_2$  can meet in one cell of the environment. For example,  $r_1$  to load  $r_2$  cannot concurrently execute services  $ld$  and  $rd$ . Unknown meeting capabilities are indicated with rotating arrows labeled with the symbol ?. For example, in Fig. 2, it is unknown whether robots  $r_1$  and  $r_2$  are able to meet in cell  $c_9$ .

An overview of MAPmAKER is depicted in 1. The *Planner* script uses the models of the robot(s) and the environment in order to compute the path-planning. The model of each robot contains information regarding its initial position, the number of services and their location that the robot must perform and the locations where the modeled robot has to synchronize with another one. The model of the environment depicts its map and defines the allowance of transitions between the cells that compound the model. This two models must be manually defined but can be reused for an infinite number of experiments. Then, the *Random model generator* generates a certain number (defined by the user) of tests based on the previously explained models. Each of this tests is unique since the generator associates a random uncertainty and initial position of the robotic team to each of them. This process is performed for each experiment, since the uncertainty that is checked changes between them (e.g. execution of transitions in Experiment 1, services provisioning in Experiment 2 and meeting capabilities in Experiment 3). The generation of models must be performed once, although we provide an already working set in our repository and this step can be skipped. This set is stored in the folder "ReplicationPackage", where the new scenarios must be saved as well.

Once the models are set, the planner can be executed selecting the suitable number of experiment and research question to be checked.

### 4 THE MAPMAKER TOOL

–Explain the tool in detail, maybe including a scope– (add figure for the tool? maybe in the previous section?)

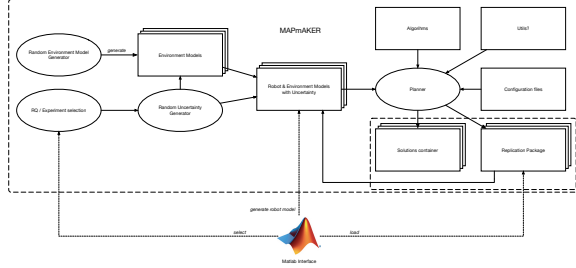


Figure 1: Overview of the MAPmAKER approach.

Our tool was developed using the MATLAB [?] environment. The tool is composed by several MATLAB scripts that can be executed independently for reaching certain functions or in a much more suitable way by means of a launcher script (/MAPmAKER/experimentLa

In the following a detailed explanation of the capabilities of the tool is presented. A set  $R = \{r_1, r_2, r_3\}$  of robots is deployed in the environment graphically described in Fig. 2. This environment represents a building made by four rooms  $L = \{l_1, l_2, l_3, l_4\}$ , which has been affected by an earthquake. The environment is further partitioned in cells, each labeled with an identifier in  $c_1, c_2, \dots, c_{30}$ . Robots  $r_1, r_2$ , and  $r_3$  are placed in their initial locations. Each robot is able to move from one cell to another, by performing action *mov*. The robots are also able to perform the following actions. Robot  $r_1$  is able to load debris of the building by performing action *ld*. In Fig. 2 the cells in which a robot  $r$  can perform an action  $\alpha$  are marked with the label  $r(\alpha)$ . Robot  $r_2$  can wait until another robot loads debris on it by performing action *rd* and can unload debris by performing one of the two actions *ud1* and *ud2*. Actions *ud1* and *ud2* use different actuators. Specifically, action *ud1* uses a gripper while action *ud2* exploits a dump mechanism. Robot  $r_3$  is able to take pictures by performing action *tp* and send them using a communication network through the execution of action *sp*. Symbols  $r_1(ld)$ ,  $r_2(rd)$ ,  $r_2(ud1)$ ,  $r_2(ud2)$ ,  $r_3(tp)$ , and  $r_3(sp)$  are used in Fig. 2 to mark the regions where actions can be executed by the robots, while movement actions are not reported for graphical reasons. Each action may be associated with a service, which is a high-level functionality provided by the robot when an action is performed. For example, actions *ld*, *rd*, *tp*, and *sp* are associated with the services *load\_carrier*, *detect\_load*, *take\_snapshot*, and *send\_info*, respectively. Actions *ud1* and *ud2* are associated with service *unload*. The labels  $L(\pi, \alpha) = \top$  below Fig. 2 are used to indicate that a service  $\pi$  is associated with action  $\alpha$ . Robots must meet and synchronously execute actions. In this example, robots  $r_1$  and  $r_2$  must meet in cell  $c_7$  and synchronously execute actions *ld* and *rd*, respectively. The cells where meeting is requested are marked with rotating arrows marked with the identifiers of the robots that must meet, meaning that, in order to meet, the robots must be on the same cell to meet.

The mission the team of robots has to achieve is to check whether toxic chemicals have been released by the container located in  $l_4$ . We assume that the mission is specified through a set of local missions assigned to each robot of the team and described in Linear Time Temporal Logic (LTL). An LTL formula is obtained

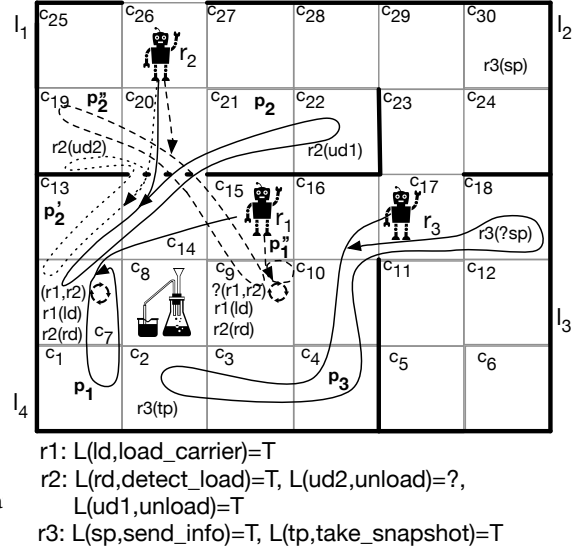


Figure 2: An example showing the model of the robots and their environment. Plans computed by MAPmAKER are represented by trajectories marked with arrows.

by composing actions with standard LTL operators:  $X$  (next),  $F$  (eventually),  $G$  (always) and  $U$  (until) [25]. In our example the mission can be specified by means of the following local missions:  $\phi_1 = G(F(load\_carrier))$ ,  $\phi_2 = G(F(detect\_load \wedge F(unload)))$ ,  $\phi_3 = G(F(take\_snapshot \wedge F(send\_info)))$ , which are assigned to robot  $r_1, r_2$  and  $r_3$ , respectively. The formulae specify that periodically robot  $r_1$  loads debris on  $r_2$  (by performing action *load\_carrier*), robot  $r_2$  receives debris (when action *detect\_load* occurs) and brings them to an appropriate unload area (by performing action *unload*), and robot  $r_3$  continuously takes pictures (by performing action *take\_snapshot*) and sends them using the communication network (by performing action *send\_info*). Informally, while  $r_3$  continuously takes pictures and sends them using the communication network,  $r_1$  and  $r_2$  remove debris to allow  $r_3$  having a better view on the container. The pictures allow verifying whether toxic chemicals have been released by the container.

## 5 CONCLUSIONS

–General conclusions (maybe use the same from the last paper but removing the discussion about the results)–

This work presented MAPmAKER, a novel decentralized planner for partially known environments. MAPmAKER solves the decentralized planning problem when partial robot applications are analyzed. The results show that the effectiveness of MAPmAKER is triggered when the computed possible plans are actually executable in the real model of the robotic application. Furthermore, in several cases, MAPmAKER was able to achieve missions that could not be completed by classical planners.

Future work and research directions include (1) the study of appropriate policies to select between definitive and possible plans,. These policies may consider the likelihood of possible plans to

be actually executable by the partial robot application, e.g., the probability that a door is open. (2) the use of more efficient planners to speed up plan computation. These may be based for example on symbolic techniques.

## REFERENCES

- [1] B. C. Akdeniz and H. I. Bozma. 2015. Exploration and topological map building in unknown environments. In *IEEE International Conference on Robotics and Automation (ICRA)*. 1079–1084.
- [2] C. Amato, G. Konidaris, G. Cruz, C. A. Maynor, J. P. How, and L. P. Kaelbling. 2015. Planning for decentralized control of multiple robots under uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*. 1241–1248.
- [3] S. Bhattacharya, R. Ghrist, and V. Kumar. 2015. Persistent Homology for Path Planning in Uncertain Environments. *IEEE Transactions on Robotics* 31, 3 (2015), 578–590.
- [4] Christos G Cassandras and Stephane LaFortune. 2009. *Introduction to discrete event systems*. Springer Science & Business Media.
- [5] Yushan Chen, Jana Tumová, and Calin Belta. 2012. LTL robot motion control based on automata learning of environmental dynamics. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 5177–5182.
- [6] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson. 2015. MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving. In *International Conference on Robotics and Automation (ICRA)*. 1670–1677.
- [7] Jonathan F Diaz, Alexander Stoytchev, and Ronald C Arkin. 2001. Exploring Unknown Structured Environments.. In *FLAIRS Conference*. AAAI Press, 145–149.
- [8] Xu Chu Dennis Ding, Stephen L Smith, Calin Belta, and Daniela Rus. 2011. LTL control in uncertain environments with probabilistic satisfaction guarantees. *IFAC Proceedings Volumes* 44, 1 (2011), 3515–3520.
- [9] Nicolás D’ippolito, Victor Braberman, Nir Piterman, and Sebastián Uchitel. 2013. Synthesizing Nonanomalous Event-based Controllers for Liveness Goals. *ACM Trans. Softw. Eng. Methodol.* 22, 1 (2013).
- [10] Noel E Du Toit and Joel W Burdick. 2012. Robot motion planning in dynamic, uncertain environments. *IEEE Transactions on Robotics* 28, 1 (2012), 101–115.
- [11] Meng Guo and Dimos V Dimarogonas. 2015. Multi-agent plan reconfiguration under local LTL specifications. *The International Journal of Robotics Research* 34, 2 (2015), 218–235.
- [12] Meng Guo, Karl H Johansson, and Dimos V Dimarogonas. 2013. Revising motion planning under linear temporal logic specifications in partially known workspaces. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 5025–5032.
- [13] Marius Kloetzer, Xu Chu Ding, and Calin Belta. 2011. Multi-robot deployment from LTL specifications with reduced communication. In *Conference on Decision and Control and European Control Conference (CDC-ECC)*. IEEE, 4867–4872.
- [14] Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. 2009. Temporal-logic-based reactive mission and motion planning. *IEEE transactions on robotics* 25, 6 (2009), 1370–1381.
- [15] Hanna Kurniawati, Yanzhu Du, David Hsu, and Wee Sun Lee. 2011. Motion planning under uncertainty for robotic tasks with long time horizons. *The International Journal of Robotics Research* 30, 3 (2011), 308–323.
- [16] Morteza Lahijanian, Matthew R Maly, Dror Fried, Lydia E Kavraki, Hadas Kress-Gazit, and Moshe Y Vardi. 2016. Iterative temporal planning in uncertain environments with partial satisfaction guarantees. *IEEE Transactions on Robotics* 32, 3 (2016), 583–599.
- [17] Jean-Claude Latombe. 2012. *Robot motion planning*. Vol. 124. Springer.
- [18] Bruno L’Espérance and Kamal Gupta. 2014. Safety hierarchy for planning with time constraints in unknown dynamic environments. *Transactions on Robotics* 30, 6 (2014), 1398–1411.
- [19] Scott C Livingston, Richard M Murray, and Joel W Burdick. 2012. Backtracking temporal logic synthesis for uncertain environments. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 5163–5170.
- [20] Savvas G Loizou and Kostas J Kyriakopoulos. 2005. Automated planning of motion tasks for multi-robot systems. In *Conference on Decision and Control and European Control Conference (CDC-ECC)*. IEEE, 78–83.
- [21] Matthew R Maly, Morteza Lahijanian, Lydia E Kavraki, Hadas Kress-Gazit, and Moshe Y Vardi. 2013. Iterative temporal motion planning for hybrid systems in partially unknown environments. In *International conference on Hybrid systems: computation and control*. ACM, 353–362.
- [22] Venkatraman Narayanan and Maxim Likhachev. 2015. Task-oriented planning for manipulating articulated mechanisms under model uncertainty. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 3095–3101.
- [23] Xinkun Nie, Lawson LS Wong, and Leslie Pack Kaelbling. 2016. Searching for physical objects in partially known environments. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 5403–5410.
- [24] Alexandros Nikou, Jana Tumova, and Dimos D Dimarogonas. 2017. Probabilistic Plan Synthesis for Coupled Multi-Agent Systems. *arXiv preprint arXiv:1704.01432* (2017).
- [25] Amir Pnueli. 1977. The temporal logic of programs. In *Foundations of Computer Science*. IEEE, 46–57.
- [26] Michael Melholt Quottrup, Thomas Bak, and RI Zamanabadi. 2004. Multi-robot planning: A timed automata approach. In *International Conference on Robotics and Automation*, Vol. 5. IEEE, 4417–4422.
- [27] Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. 2006. Planning under uncertainty for reliable health care robotics. In *Field and Service Robotics*. Springer, 417–426.
- [28] Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. 2006. *Planning under Uncertainty for Reliable Health Care Robotics*. Springer, 417–426.
- [29] Philipp Schillinger, Mathias Bürger, and Dimos Dimarogonas. 2016. Decomposition of Finite LTL Specifications for Efficient Multi-Agent Planning. In *International Symposium on Distributed Autonomous Robotic Systems*.
- [30] Wolfgang Thomas et al. 2002. *Automata, logics, and infinite games: a guide to current research*. Vol. 2500. Springer Science & Business Media.
- [31] Jana Tumova and Dimos V Dimarogonas. 2016. Multi-agent planning under local LTL specifications and event-based synchronization. *Automatica* 70 (2016), 239–248.
- [32] Eric M Wolff, Ufuk Topcu, and Richard M Murray. 2012. Robust control of uncertain Markov decision processes with temporal logic specifications. In *Annual Conference on Decision and Control (CDC)*. IEEE, 3372–3379.
- [33] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M Murray. 2009. Receding horizon temporal logic planning for dynamical systems. In *Conference on Decision and Control*. IEEE, 5997–6004.