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, MAPmAKERtackle 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

MAPmAKERprovides 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 [18]. 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 [14, 21, 27] and that is decomposed into a set of *local missions*[11, 11, 30, 33]. Every robot is commanded to achieve a local mission, specified as a LTL property. As seen in [33], 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, 28]. 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. ROS Moveit http://moveit.ros.org/, Gazebo http://gazebosim.org/ or look for papers presenting tools*◀

Decentralized planning problem has been studied for known environments [11, 30, 33]. However, planners for partially known environments do not usually employ decentralized solutions [7, 10, 28].

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, 16, 23, 25, 29, 34]) 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 [34], transition probabilities are not exactly known but are known to belong to a given uncertainty sets. Finally, several works [16, 29] 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, 17, 19, 20, 24]) 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 [17], 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

ICSE 2018, Gothenburg, Sweden

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, 15, 20, 35]) 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, 20, 32] techniques and iterative planning [12, 22]. 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. MAPmAKER modifies [33] by supporting partial knowledge. This tool splits the given set of robot that conforms the team into classes depending on the local mission that each of them must achieve. Then, the planner computes possible and definitive plans based on the model of the environment and its partial knowledge. 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. If MAPmAKERis able to find a plan, it is performed in order to achieve each robot's mission. The tool always try to reach the goal performing the lower number of actions.

Each robot is able to perform a complex mission, therefor the goals that they reach in the environment represent points where a service is provided Sergio ►*reference to the other paper containing the running example?*◄ or where two robots can synchronize. However, in order to reach these goals robots must face different issues related with partial knowledge, as listed in the following lines.

Partial knowledge about the actions execution. The transition between two of the cells that conform the grid map of the environment can be: (1) always possible; (2) always not possible (i.e. a wall); (3) not known (i.e. a door between two rooms that can be open or closed). MAPmAKERis able to compute a plan that goes through an unknown transition, therefor performing a *possible plan*.

Unknown service provisioning. In the dynamic environments studied for this work whether a service is provided or not in a specific location could be unknown. Sergio ►*reference to the other paper containing the running example?*◄ In order to achieve the mission with the minimum number of actions performed, MAPmAKERis able to compute a plan that tries to reach an uncertain service. Thus, MAPmAKERcomputes a *possible plan*.

Unknown meeting capabilities. As stated before, robots can meet and synchronize in certain locations. There they can perform a mission, as exchanging a load, in a collaborative fashion. Sergio ►*reference to the other paper containing the running example?*◄ In this case, two or more robots that are part of the same subset part of the whole team can compute a plan whose goal is to meet in an uncertain synchronization point. Thus, MAPmAKERcomputes a *possible plan*.

Whenever a robot approaches to an location with uncertain information, it detects if this transition, service, or meeting capability is detected to be firable, provided, and possible, respectively. The robot then updates the information concerning to this location, setting it yo "true" or "false". This information is shared with the rest of the team so it can be take into account for further planning.

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 that the robot must perform and their location and the locations where the modeled robot has to synchronize with another one. The model of the environment represents its map and defines the allowance of transitions between the cells that compound the model (i.e. walls). 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. The already existing models of the environment represent models of real robot applications, i.e., the RoboCup Logistics League competition [13] and an apartment of a large residential facility for senior citizens [31]. 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 number of experiment to perform and the research question to be checked. The approach that is followed in order to test the uncertainty of the environment and its repercussion into the performance of the tool is explained in the following.

Step 1. For the first step a partial model of the whole robot application is loaded. MAPmAKERthen computes a *possible plan*, which is executed by the robot(s). When executing the plan, every time a robot detects a false evidence about a partial information

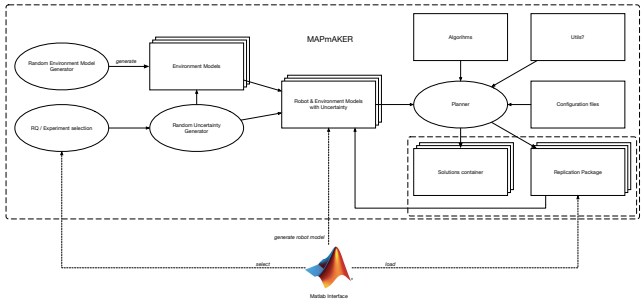

Figure 1: Overview of the MAPmAKER approach.

, e.g., a transition of the plan is not executable, MAPmAKERis re-executed to recompute a new *possible plan.*

Step 2. For this case all the unknown information regarding the robotic application is set to false, meaning that MAPmAKERwill always return a *definitive plan.*

During the performance of the two steps for a certain number of experiments MAPmAKERcollects information regarding: (1) the length of each plan; (2) the time that took to the tool to compute each plan; (3) the ratio between the length of definitive and possible plans; (4) the ratio between the time expect computing definitive and possible plans; (5) the false and true evidences found during execution.

4 THE MAPMAKER TOOL

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

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/experimentLauncher.m).

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 $c_1, c_2, \ldots, c_{30}$. 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 α 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 *ud*1 and *ud*2. Actions *ud*1 and *ud*2 use different actuators. Specifically, action *ud*1 uses a gripper while action *ud*2 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

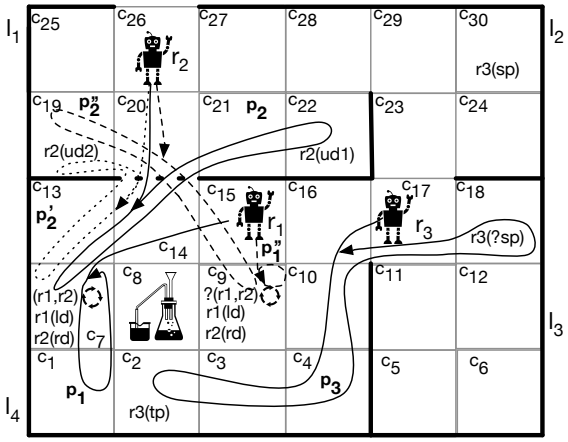

Figure 2: An example showing the model of the robots and their environment. Plans computed by MAPmAKER are represented by trajectories marked with arrows.

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 *ud*1 and *ud*2 are associated with service *unload*. The labels $L(\pi, \alpha) = \top$ below Fig. 2 are used to indicate that a service π is associated with action α. 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 by composing actions with standard LTL operators: X (next), F (eventually), G (always) and U (until) [26]. In our example the mission can be specified by means of the following local missions: $\phi_1 = \mathrm{G}(\mathrm{F}(load\_carrier))$, $\phi_2 = \mathrm{G}(\mathrm{F}(detect\_\ load \wedge \mathrm{F}(unload)))$, $\phi_3 = \mathrm{G}(\mathrm{F}(take\_snapshot \wedge \mathrm{F}(send\_info)))$, 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] Christian Deppe Ulrich Karras, Tobias Neumann, Tim Niemueller Alain Rohr, Wataru Uemura, Daniel Ewert, Nils Harder, Sören Jentzsch, Nicolas Meier, and Sebastianyear Reuter. 2016. RoboCup Logistics League Rules and Regulations. (2016).

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Jean-Claude Latombe. 2012. *Robot motion planning*. Vol. 124. Springer.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] Alexandros Nikou, Jana Tumova, and Dimos D Dimarogonas. 2017. Probabilistic Plan Synthesis for Coupled Multi-Agent Systems. *arXiv preprint arXiv:1704.01432* (2017).

[26] Amir Pnueli. 1977. The temporal logic of programs. In *Foundations of Computer Science*. IEEE, 46–57.

[27] 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.

[28] Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. 2006. Planning under uncertainty for reliable health care robotics. In *Field and Service Robotics*. Springer, 417–426.

[29] Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. 2006. *Planning under Uncertainty for Reliable Health Care Robotics*. Springer, 417–426.

[30] 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*.

[31] The Angen research and innovation apartment: official website 2014. (2014). http://angeninnovation.se

[32] Wolfgang Thomas et al. 2002. *Automata, logics, and infinite games: a guide to current research*. Vol. 2500. Springer Science & Business Media.

[33] Jana Tumova and Dimos V Dimarogonas. 2016. Multi-agent planning under local LTL specifications and event-based synchronization. *Automatica* 70 (2016), 239–248.

[34] 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.

[35] 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.