Multi-Agent Path Finding on Real Robots

Demonstration

Roman Barták, Ivan Krasičenko, Jiří Švancara
Charles University, Faculty of Mathematics and Physics
Praha, Czech Republic
bartak@ktiml.mff.cuni.cz

ABSTRACT

Multi-agent path finding (MAPF) deals with the problem of finding a collision-free path for a set of agents in a graph. It is an abstract version of the problem to coordinate movement for a set of mobile robots. This demo presents software guiding through the MAPF task, starting from the problem formulation and finishing with execution of plans on real robots. Users can design grid-like maps, specify initial and goal locations of robots, generate plans using various abstract models implemented in the Picat programming language, simulate and visualize execution of these plans, and translate the plans to command sequences for Ozobots, small robots developed for teaching programming.

KEYWORDS

path planning; multi-robot; Ozobot

ACM Reference Format:
Roman Barták, Ivan Krasičenko, Jiří Švancara. 2019. Multi-Agent Path Finding on Real Robots. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 3 pages.

1 INTRODUCTION

Abstraction is the process of removing details from a problem representation. It is a critical step in problem solving as without abstraction "intelligent agents would be completely swamped by the real world" [7]. Despite its importance, little attention has been paid to abstraction techniques compared to, for example, solving techniques. In this demo, we look at a specific planning problem called *multi-agent path finding* (MAPF) that deals with finding collision-free paths for a set of agents. MAPF has practical applications in video games, traffic control, and robotics (see Felner et al. [4] for a survey). There exists a widely-accepted uniform abstract model of MAPF consisting of an undirected graph describing allowed locations and movements of agents and two possible abstract actions: *move* for moving to a neighbouring node and *wait* for waiting at the current node. The research question is if this abstract model is appropriate for problems with real robots and, if not, how the abstract model of MAPF should look like.

We present software for experimental evaluation of various abstract models by executing the obtained plans on real robots. The software provides a visual editor to state the MAPF problems on a grid map, interface for solvers of the MAPF problems written in the Picat language, visualisation of plans and plan execution, transformation of plans to control procedures for the Ozobot robots, and tools supporting execution of plans. The software is intended as a research tool for testing various abstract models of the MAPF problem on real robots. The initial results comparing several abstract models were already published [2].

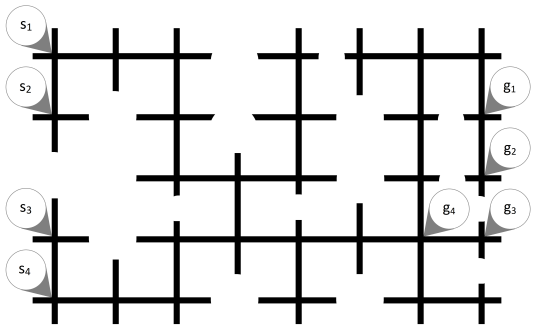

Figure 1: A grid map for MAPF. Agents follow the black line, the gray circles indicate starting and goal locations.

2 BACKGROUND ON MAPF

The MAPF problem is defined by a graph $G = (V, E)$ and a set of agents $a_1, \ldots, a_k$, where each agent a_i is associated with starting location $s_i \in V$ and goal location $g_i \in V$. A grid map with a unit length of each edge is often used to represent the environment [8], Figure 1 shows an example of such a map. The task is to find a collision-free path for each agent from its starting location to its goal location. There exist versions of the MAPF problem, for example, the k-robust version, that is particularly interesting for real robots as the plans are supposed to be robust to possible delays during execution. Formally, k-robust plans require for each vertex of the graph to be unoccupied for at least k time steps before another agent can enter it [1]. Perhaps due to many practical applications in areas such as automated warehouses, interest in MAPF increased in recent years and many solving techniques have been proposed. Our system uses a reduction-based solver in the Picat programming language [3] as it is easy to encode versions of the MAPF model there. Reduction-based solvers exploit the idea of translating the problem to another formalism and solving the problem using the techniques developed for that formalism. Picat solves the MAPF problem by translating it to SAT, which is currently the most popular reduction-based approach for MAPF, though other solvers such as CP (constraint programming) and IP (integer programming) can also be used.

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

The abstract plan outputted by MAPF solvers is a sequence of locations that the agents visit (or equivalently a sequence of move and wait operations). Before execution on a real robot, the abstract plan needs to be translated to a sequence of actions that the physical robot can perform. Our system supports the Ozobot robots [6], see Figure 2, that provide high-level actions such as turn left and right and move forward so it is not necessary to deal with low-level control. By concatenating these actions, the agent can perform all the required steps from the abstract plan. This translates to five possible actions at each time step - (1) wait, (2) move forward, (3,4) turn left/right and move, and (5) turn back and move. As the mobile robot cannot move backward directly, turning back is implemented as two turns right (or left).

As the abstract steps may have duration different from the physical steps, the abstract plans, which are perfectly synchronized, may desynchronize when being executed, which may further lead to collisions. This is even more probable in dense and optimal plans, where agents often move close to each other. The intuition says that such desynchronization will indeed happen. The presented software has beed designed to verify this hypothesis by providing tools to evaluate various abstract models and various translations of abstract actions to executable actions. Note that the used robots only blindly follow the computed plan and cannot intervene if, for example, an obstacle is detected.

3 SYSTEM CAPABILITIES

The presented system supports the whole process of solving MAPF problems. The user can define a grid map, put obstacles there by removing vertices and edges, and specify initial and goal locations of agents. The map can be printed for usage with Ozobots or it can be displayed on the computer screen and robots can move on the screen directly. The system provides encodings of several MAPF models in the Picat programming language including the classical model [3], the 1-robust model [1], and a model that includes turning actions in addition to move and wait actions [2]. There is also an interface for adding other models. Problem solving can be directly realized from the software, which generates the problem specification for the solver from the map drawn by the user. The generated plans can then be visualized as a timeline of actions for each robot (Gantt chart). The system can also visualize execution of plans. Finally, the plans can be exported for execution on Ozobots; the system allows users to specify durations of actions for execution. As we already mentioned, the robots can be then placed on a printed map to execute the plans (the map can be printed from the application) or the robots can run on the computer screen with the map displayed there. In this second case, the system also shows where the robots are supposed to be so the user can see how the real plan execution corresponds to expected execution. Figure 3 shows the integrated user interface of the software. Video presenting the system is available at [9].

Figure 2: Ozobot Evo from Evollve. Picture is taken from [6].

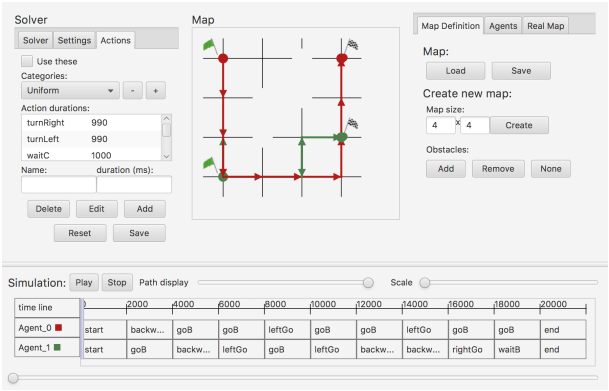

Figure 3: User interface of the MAPF system.

4 CONCLUSIONS AND FUTURE STEPS

The presented system is intended to study various abstract models of the MAPF problem from the perspective of plan execution on real robots Ozobots. The initial empirical evaluation [2] showed that there is indeed a gab between widely-used theoretical frameworks for MAPF and deployment of solutions in real environments. A wider experimental study is necessary to understand better the relations between abstract models and real environments. For example, the ratio between the length of edges and the size of robots is important as longer edges diminish the influence of turning time and also put space between the robots, which decreases the chance of collision. The presented system allows users to define the length of edges so such studies can be realised in future. Similarly, the system allows users to define own abstract models of MAPF so other abstractions can be studied in future. Currently, blind execution of plans is assumed, which means that sensors are not used during execution. It would be interesting to look at plan-execution policies that assume communication between agents and exploit information from sensors [5]. The system allows users to modify the execution strategy by using different command sequences so more advanced execution strategies can be implemented in future.

The presented system provides to the MAPF community a tool for bridging the abstract models and plan execution on real robots. Thanks to using a standard platform of Ozobots, no specific expertise in robotics is necessary.

Acknowledgements

Research is supported by the Czech Science Foundation under the project P103-19-02183S and by the Charles University Grant Agency under the project 90119.

REFERENCES

[1] Dor Atzmon, Ariel Felner, Roni Stern, Glenn Wagner, Roman Barták, and Neng-Fa Zhou. 2017. k-Robust Multi-Agent Path Finding. In *Proceedings of the Tenth International Symposium on Combinatorial Search (SoCS)*. AAAI Press, Palo Alto, 157–158. https://aaai.org/ocs/index.php/SOCS/SOCS17/paper/view/15797

[2] Roman Barták, Jiří Švancara, Věra Škopková, and David Nohejl. 2018. Multi-agent Path Finding on Real Robots: First Experience with Ozobots. In *Advances in Artificial Intelligence - IBERAMIA 2018 - 16th Ibero-American Conference on AI, Trujillo, Peru, November 13-16, 2018, Proceedings*, Guillermo R. Simari, Eduardo Fermé, Flabio Gutiérrez Segura, and José Antonio Rodríguez Melquiades (Eds.). Springer International Publishing, Cham, 290–301. https://doi.org/10.1007/978-3-030-03928-8_24

[3] Roman Barták, Neng-Fa Zhou, Roni Stern, Eli Boyarski, and Pavel Surynek. 2017. Modeling and Solving the Multi-agent Pathfinding Problem in Picat. In *29th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE Computer Society, 959–966. https://doi.org/10.1109/ICTAI.2017.00147

[4] Ariel Felner, Roni Stern, Solomon Eyal Shimony, Eli Boyarski, Meir Goldenberg, Guni Sharon, Nathan R. Sturtevant, Glenn Wagner, and Pavel Surynek. 2017. Search-Based Optimal Solvers for the Multi-Agent Pathfinding Problem: Summary and Challenges. In *the International Symposium on Combinatorial Search (SoCS)*. AAAI Press, Palo Alto, 29–37.

[5] Hang Ma, T. K. Satish Kumar, and Sven Koenig. 2017. Multi-Agent Path Finding with Delay Probabilities. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*. AAAI Press, Palo Alto, 3605–3612.

[6] Ozobot & Evollve, Inc. 2018. *Ozobot | Robots to code, create, and connect with.* https://ozobot.com/

[7] Stuart J. Russell and Peter Norvig. 2009. *Artificial Intelligence: A Modern Approach.* Prentice Hall.

[8] Malcolm Ross Kinsella Ryan. 2008. Exploiting Subgraph Structure in Multi-Robot Path Planning. *J. Artif. Intell. Res.* 31 (2008), 497–542. https://doi.org/10.1613/jair.2408

[9] Jiří Švancara and Ivan Krasičenko. 2019. *MAPF Scenario video demo.* Charles University. https://drive.google.com/file/d/1mkHq_Xgkp8QH3H60I7R27g4Q6RWzIfY7/view