OzoMorph: Demonstrating Colored Multi-Agent Path Finding on Real Robots

Roman Barták, Jakub Mestek
Charles University, Faculty of Mathematics and Physics
Prague, Czech Republic
bartak@ktiml.mff.cuni.cz

Abstract

Multi-agent path finding (MAPF) deals with finding collision-free paths for a set of agents on a graph, where each agent has its origin and destination. Colored MAPF is a generalization of MAPF, where groups of agents are moving and the set of destination nodes is specified per group rather than per agent. OzoMorph is software providing intuitive user interface for specifying instances of Colored MAPF problems, solving them by translation to SAT, and finally visualizing the solution either in computer simulation or by converting the plans to executable instructions for Ozobot robots.

Introduction

Multi-agent Path Finding is a growing research area dealing with problems of finding collision-free paths for a set of agents – robots (Stern et al. 2019). The problem has numerous applications in areas such as automated warehousing, parking, airplane taxiing, and computer games.

The environment, in which the robots move, is usually abstracted as an un-directed graph, where robots wait in nodes and move over the edges to reach their destination. Frequently, this graph is a 4-connected grid as it can naturally cover various maps and has the property of equally long edges, which simplifies synchronization of agents. The task is to find a path for each agent from its start node to its destination node. The paths should be collision free, which means that robots should not crash to each other. There are various formal descriptions of collisions (Stern et al. 2019), the most widely used are *vertex collisions* (agents are at the same vertex at the same time) and *swapping collisions* (agents swap their positions using the same edge). As robots might be late when executing their plans, k*-robust plans* were introduced that guarantee collision-free execution even if any agent is late for at most k actions (Atzmon et al. 2020). The usual abstract actions used in plans are *move* (to a neighboring node) and *wait* (in a given node). These actions are supposed to have identical duration to ensure that robots stay synchronized during execution. This may however prolong the plan execution as some actions (such as moving forward) are naturally faster than others (such as turning and then moving forward). Therefore models with variable action duration (Barták, Švancara, and Vlk 2018) or with a extended set of actions (for example, by turning actions) (Barták et al. 2019) were proposed.

In this paper, we focus on generalization of MAPF called *Colored MAPF* (Solovey and Halperin 2014), where agents are grouped and destinations are specified for each group rather than for individual agents (if groups are singleton then we get classical MAPF). Specifically, we present software OzoMorph that demonstrates solving Colored MAPF and executing the plans on robots called Ozobots (Evollve, Inc. 2018). The software allows users to easily define the problem. We use rectangular 4-connected grids for maps so users just need to draw initial and goal positions of agent groups. The problem is then solved by translation to SAT via the Picat system (Zhou and Kjellerstrand 2016). We use the model with move, turn and wait actions with 1-robustness to ensure smooth execution of plans on robots (Barták et al. 2019), but any model can be developed by users in Picat. Finally, the found plans are demonstrated either in simulation on a computer screen or on real robots, where the system generates control instructions for Ozobots.

Colored MAPF

Let $A = \{1, \ldots, n\}$ be a set of n agents and $G = (V, E)$ be an un-directed graph. Agents are initially staying at some nodes, which is described by initial configuration $S : A \rightarrow V$, where $S(a)$ is the initial position of agent a. The final configuration is given by a set of nodes $T \subset V$ such that $|T| = |A|$. *Anonymous MAPF* problem is given by a quadruple (G, A, S, T) and its solution is a set of collision-free plans. A plan π_a for an agent a is a sequence of vertices such that $\pi_a[1] = S(a)$ and for each t either $\pi_a[t] = \pi_a[t+1]$ (agent waits at vertex) or $(\pi_a[t], \pi_a[t{+}1]) \in E$ (agent moves to a neighboring vertex). Let m_a be the length of plan for agent a, then we define $\pi_a[t] = \pi_a[m_a]$ for each $t > m_a$ (agents stay in their final nodes). Let $mks = max_{a \in A}\ m_a$ be a makespan of the plans. We require agents to reach the final configuration: $\forall v \in T\ \exists a \in A : \pi_a[mks] = v$, and the plans to be collision free: $\forall a_1, a_2 \in A, a_1 \neq a_2, \forall t : \pi_{a_1}[t] \neq \pi_{a_2}[t]$ (no vertex collision) and $\forall a_1, a_2 \in A, a_1 \neq a_2, \forall t : \pi_{a_1}[t] \neq \pi_{a_2}[t+1] \vee \pi_{a_1}[t+1] \neq \pi_{a_2}[t]$ (no swapping collision).

Colored MAPF (also called Team MAPF or TAPF) with k groups is then given as $(G, (A_1, S_1, T_1), \ldots, (A_k, S_k, T_k))$, where each (G, A_i, S_i, T_i) is anonymous MAPF (Solovey and Halperin 2014). A solution of Colored MAPF is union of solutions of individual anonymous MAPF problems such that the plans across the groups are also collision-free.

Anonymous MAPF can be solved makespan-optimally in polynomial time (Yu and LaValle 2012), but finding makespan-optimal solution to Colored MAPF is NP-hard (Surynek 2010) (as Colored MAPF is a generalization of classical MAPF, where each agent has own destination). There exists an optimal algorithm for solving Colored MAPF, called Conflict-Based Min-Cost-Flow (Ma and Koenig 2016), but in this work we decided to adapt a SAT-based approach for two reasons. First, the model for classical MAPF (Barták et al. 2017) can be easily modified to solve Colored MAPF (only the final condition is changed). Second, this approach gives us large flexibility in exploring different models. In particular, we added constraints for 1-robustness and we used finer-resolution actions that also included turning. This gives us plans that are more robust during execution on robots (Barták et al. 2019).

OzoMorph Software

OzoMorph is a Java program for formulating Colored MAPF problems, their solving by translation to SAT, and demonstrating the plans either by simulation on a computer screen or by execution on robots Ozobot Evo. The user first enters the size of the 4-connected grid map and then freely allocates "robots" to their initial positions in one map and final positions in the other map (see Figure 1). Robots are identified by colors so the above formulation corresponds to Colored MAPF problem. Artistically speaking, the user paints two pixel-based color pictures, where the number of pixels of the same color at both pictures must be identical. The software indicates these numbers so it is easy to spot if some agents are missing in either picture. Then, by pressing a single button, the software generates the problem instance for the Picat-based solver and solves it. As described above, we use the model with move, turn, and wait actions and with 1-robustness constraint. Advanced users may write own models in the Picat programming language.

Obtained plans can then be executed in simulation on a computer screen or the system can generate control codes for Ozobot Evo robots. These codes are then be uploaded to individual robots (usually via Bluetooth). The system supports two modes of execution that differ in how the synchronized start of all robots is realized. In one mode, the robots run on a horizontally-placed computer screen (or a tablet) that shows the map (Figure 2) and starts all the robots synchronously by displaying a specific color on the screen. In the second mode, the robots run on a paper map, that can be printed from the application. Then robots must be started one by one at specified times (the application provides the timer) to ensure that the plans start synchronously.

When translating the abstract plans obtained from the Picat solver, the system uses predefined templates for actions written in the Ozoblockly language (Evollve, Inc. 2015) – the visual language for programming Ozobots (Figure 3). Advanced users may redefine these templates to use different control sequences for actions or a different approach to start agents.

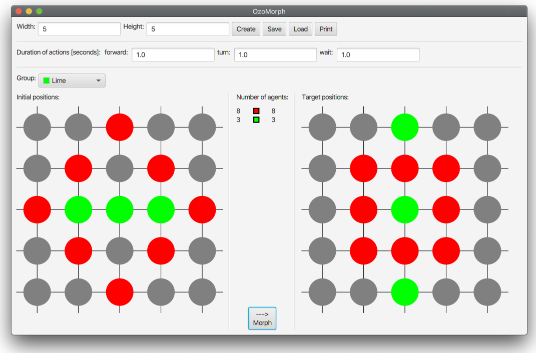

Figure 1: OzoMorph user interface.

Figure 2: Ozobot Evo robots on a screen displaying the map.

Conclusions

The OzoMorph software supports the whole development cycle for Colored MAPF, starting with visual definition of problem instance, through solving the problem, and finishing with translation of plans to robot control sequences or simulating the plans. It can be used for artistic purposes to generate programs morphing one configuration of robots (picture) to another configuration or for research purposes to compare different SAT-based models of the problem and different translations to robot control sequences.

Acknowledgments. This work is supported by the project P103-19-02183S of the Czech Science Foundation.

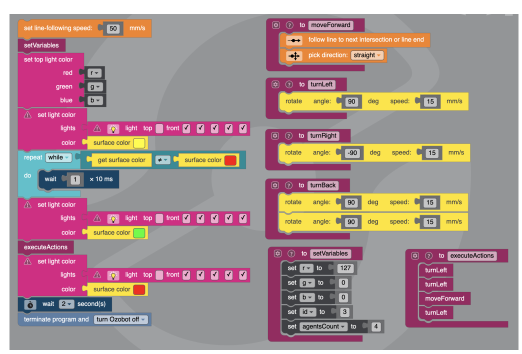

Figure 3: Example of Ozoblockly code for one robot.

References

Atzmon, D.; Stern, R.; Felner, A.; Wagner, G.; Barták, R.; and Zhou, N. 2020. Robust Multi-Agent Path Finding and Executing. *J. Artif. Intell. Res.* 67: 549–579. doi:10.1613/jair.1.11734. URL https://doi.org/10.1613/jair.1.11734.

Barták, R.; Švancara, J.; Škopková, V.; Nohejl, D.; and Krasičenko, I. 2019. Multi-agent path finding on real robots. *AI Communications* 32(3): 175–189. doi:10.3233/AIC-190621. URL https://doi.org/10.3233/AIC-190621.

Barták, R.; Švancara, J.; and Vlk, M. 2018. A Scheduling-Based Approach to Multi-Agent Path Finding with Weighted and Capacitated Arcs. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, 748–756. URL http://dl.acm.org/citation.cfm?id=3237494.

Barták, R.; Zhou, N.; Stern, R.; Boyarski, E.; and Surynek, P. 2017. Modeling and Solving the Multi-agent Pathfinding Problem in Picat. In *29th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2017, Boston, MA, USA, November 6-8, 2017*, 959–966. doi:10.1109/ICTAI.2017.00147. URL https://doi.org/10.1109/ICTAI.2017.00147.

Evollve, Inc. 2015. *Welcome to OzoBlockly*. https://ozoblockly.com/.

Evollve, Inc. 2018. *Ozobot — Robots to code, create, and connect with*. https://ozobot.com/.

Ma, H.; and Koenig, S. 2016. Optimal Target Assignment and Path Finding for Teams of Agents. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, May 9-13, 2016*, 1144–1152. URL http://dl.acm.org/citation.cfm?id=2937092.

Solovey, K.; and Halperin, D. 2014. *k*-color multi-robot motion planning. *I. J. Robotics Res.* 33(1): 82–97. doi:10.1177/0278364913506268. URL https://doi.org/10.1177/0278364913506268.

Stern, R.; Sturtevant, N. R.; Felner, A.; Koenig, S.; Ma, H.; Walker, T. T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. K. S.; Barták, R.; and Boyarski, E. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *Proceedings of the Twelfth International Symposium on Combinatorial Search, SOCS 2019, Napa, California, 16-17 July 2019*, 151–159. URL https://aaai.org/ocs/index.php/SOCS/SOCS19/paper/view/18341.

Surynek, P. 2010. An Optimization Variant of Multi-Robot Path Planning Is Intractable. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1768.

Yu, J.; and LaValle, S. M. 2012. Multi-agent Path Planning and Network Flow. In *Algorithmic Foundations of Robotics X - Proceedings of the Tenth Workshop on the Algorithmic Foundations of Robotics, WAFR 2012, MIT, Cambridge, Massachusetts, USA, June 13-15 2012*, 157–173. doi:10.1007/978-3-642-36279-8_10. URL https://doi.org/10.1007/978-3-642-36279-8_10.

Zhou, N.; and Kjellerstrand, H. 2016. The Picat-SAT Compiler. In *Practical Aspects of Declarative Languages - 18th International Symposium, PADL 2016, St. Petersburg, FL, USA, January 18-19, 2016. Proceedings*, 48–62. doi:10.1007/978-3-319-28228-2_4. URL https://doi.org/10.1007/978-3-319-28228-2_4.