Some Thoughts on Robustness in Multi-Agent Path Finding

Roman Barták

Charles University, Faculty of Mathematics and Physics
Malostranské náměstí 25, 118 00 Prague, Czech Republic
bartak@ktiml.mff.cuni.cz

Abstract

Multi-agent path finding deals with finding collision free paths for a group of agents moving to given destinations. The off-line generated plan is assumed to be blindly executed on robots, which brings issues when something is not going according to the plan. This short paper discusses robustness as a way to prevent the issues with uncertainty, dynamicity, and possible involvement of other (uncontrolled) agents.

Background and Assumptions

Multi-agent path finding (MAPF) is a problem of finding collision free paths for a set of agents moving usually on a graph (Stern et al. 2019). Each agent is given initial and goal destinations (vertices) and the task is to find a path (in the graph) that does not collide with paths of other agents. Collision usually means that two agents are present at the same time at the same location (vertex or edge). MAPF problem is the core problem in several areas with automated warehouses (Wurman et al. 2008) as one of the key applications.

As the agents in warehouses are frequently controlled by a single entity, centralized solving approaches are prevailing in the MAPF community. This makes it easier to guarantee collision-free paths and optimal solutions with respect to various objectives (such as makespan and sum of individual costs). On the other hand, it expects perfect execution of plans and no external disturbances, e.g. from other agents. Plus, there is a scalability issue due to hardness of the MAPF problem (Ratner and Warmuth 1990; Yu and LaValle 2013; Surynek 2015). Even without external influence, the imprecision of hardware (robots) executing the plans, such as delay of robots, and not fully accurate problem abstraction, such as ignoring some real actions (localisation, rotations etc.), causes that the collision-free abstract plans may lead to collision during execution (Barták et al. 2019). This gap between real and abstract worlds may be solved at the level of execution, for example, by robust execution strategies, or at the level of problem solving by using abstraction tolerant to execution inaccuracies, for example, by generating robust plans. In this short paper, we summarise the major approaches to robustness and sketch some ideas for future development towards real-life environments.

As warehouse robots are a very important application area for MAPF, we will use assumptions derived from that environment. Warehouse robots are used in massive quantities, and hence low production cost is expected. This is reflected in some limitations of hardware platforms used. First, there are limited capabilities of sensors. We will assume that there are sensors for localisation and for low-distance obstacle detection. Second, there are limited communication capabilities. We will assume that robots can communicate with central entity but not directly among themselves. Last but not least, there is only low-performance computing equipment onboard of robots, which focuses mainly on execution of known commands and reactive response to external events. As an abstraction level for planning, we assume a graph with uniform length of edges, such as a grid map, which is typical in MAPF. We assume that robots can localise themselves on that graph (but localisation may take time) and can move over the edges (but speed may vary). Other uncontrolled agents (not under the control of central entity), such as humans may appear in the environment. As the controlled agents cannot communicate with other agents directly, we can model the uncontrolled agents as dynamic obstacles in the environment. The limited sensoric capabilities of controlled agents allows that, but, for example, reasoning about other agents is beyond their capabilities.

Robust Execution

A classical MAPF algorithm generates plans for agents consisting of sequences of vertices to visit (wait operation is reflected by repeating the same vertex in the sequence). In one time step, all robots synchronously execute the action of moving to the next vertex[1]. We assume that robots move only when the next vertex is not occupied (low-distance obstacle detection covers that). If there is an obstacle, the robot simply waits, which is reflected in delay. The central entity is always informed about the locations of agents and distributes go-on (continue in the plan) or wait instructions.

There exists well-accepted concept that translates abstract plans to plan-executable strategies and guarantees collision-free execution (Hönig et al. 2016). The idea is converting the abstract plan to a temporal plan graph that describes se-

[1]Other operations, such as rotation, can also be covered by this abstraction (Barták et al. 2019).

quence of nodes to visit for each agent and also interaction of agents in nodes reflected in a temporal constraint specifying which agent is going first in a given node. This graph is basically a simple temporal network that allows scheduling of agents (assigning times to move actions) under various kinematic constraints, but also allows robust execution of plans. Robot is allowed to move to a given node only when all the robots supposed to move through that node earlier have already passed through the node. This guarantees collision-free execution of plans even under delays of agents and hence we call it *robust execution*. Note that this approach handles also uncontrolled agents as they simply block the controlled agents and cause their delay. Hence the notion of *delay* seems to be a uniform model to handle uncertainty of execution and dynamicity of environment including uncontrolled agents. Too long delay (new permanent obstacle) may lead to re-planning by the central entity.

The drawback of above approach is that an agent may wait for the preceding agent for a long time (even infinitely long, if that preceding agent is broken), which may negatively influence objectives (delayed arrival of agent). Recently, a simple cure has been proposed using switchable temporal plan graph (Paul, Feng, and Li 2023), which is basically a disjunctive temporal network (DTN). The idea is that agents may switch the order in which they go through a given node (switch with respect to the original plan) and hence allow the originally later agent to continue through the node, even if the previous agent is delayed. One should, however, realise that checking consistency of DTN is an NP-hard problem (Stergiou and Koubarakis 2000), so allowing full re-scheduling (selecting/changing of order of agents in future nodes) may be time consuming and not appropriate for real-time execution.

Robust Solutions

Robust execution allows a MAPF plan to be executed in real-life environment by responding to unexpected situations that are reflected as delays of agents. It is a reactive approach that responses fast to the current situation.

Another approach is proactive generating plans that are robust. It means that the same plan can be executed even if there are some modifications of the environment. Again, various unexpected situations that may happen are modelled as delay of agents. The concept of k-*robustness* has been proposed to formally express robustness under delay (Atzmon et al. 2020). The plan is k-robust if it can be executed without any change even if any agent is delayed at most k steps. One can imagine such a plan as each agent having a bubble around where no other agent can be present at any time of the plan. k-robust plans always assume the worst-case situation and hence quality of plans degrades with increasing k and plans may even not exist if the environment is too dense.

Humans use a less demanding approach to robustness that we can describe as "having plan B". This idea has been applied to MAPF problems (Nekvinda and Barták 2021) in the following way. Given a plan solving the MAPF problem, we can identify possible collision points – these are basically vertices being crossed by multiple agents. For a given agent and vertex, we calculate time intervals when this agent is forbidden to be at the vertex (as other agents are supposed to be there). Now, if the agent is delayed in a previous vertex such that the agent would enter the next vertex in a forbidden interval, we can divert that agent to a different path. So rather than letting the agent on a path possibly colliding with another agent, we divert the agent to another collision-free path. This is similar to contingency planning and it is computationally more demanding than finding a single plan. The existing approach uses a main plan for each agent and alternative plans in case of delay in some vertices (but no further alternative on alternative plans are assumed and alternative plans are collision-free with main plans of other agents only, but not with their alternative plans). Chances to find such plans with alternatives are higher than finding k-robust plans while having similar practical robustness during execution.

Next Steps

In the previous sections with discussed several existing approaches for executing MAPF plans in environments, where not everything runs as expected. We looked for a property called *robustness* both at the execution level and at the planning level. We briefly sketched that having "better" execution is connected with worse scalability. Now, we will look at existing approaches that might be applied there but as far as we know, have not been tried within MAPF yet.

The above-discussed robust execution depends on a central entity distributing commands to individual agents. This brings some guarantees (for example, preventing deadlocks), but has the scalability issues. In distributed environments, more responsibility is put on individual agents but this may go against the assumption of limited capabilities of agents. There already exists a concept of *social rules* used in multi-agent planning (Karpas, Shleyfman, and Tennenholtz 2017). Executing such rules should not be computationally demanding but the open question is what the rules should be for MAPF (will classical traffic rules work there?). As these rules work locally, they may lead to dead-ends, but the centralised planner generating the original plans may prepare the plans to minimise chances of such situations.

Plans are in principle rigid and they will always have issues with unexpected situations, which need to be solved by re-planning. Contingency planning can be seen as preparing alternative plans off-line rather then re-planning, but to be practically efficient, the alternative plans need to be "sparse" as it is not efficient to generate plans for situations occurring with low probability. There exist a flexible concept of Markov Decision Process (MDP) that uses policies (what to do in each situation) rather than plans. It would fit well the limited capabilities of the hardware for execution, but it is very computationally demanding if applied directly. The idea could be to apply MDP for each agent to restricted set of nodes along the path to destination similarly to exploring limited paths when checking collisions with other agents (Svancara et al. 2023).

Policies for individual agents may not cover all collisions due to complexity issues, but these collisions might be handled by the robust execution system. In summary, tighter integration of execution strategy with planning may bring a scalable solution with some robustness guarantees.

Acknowledgments

Research is supported by the Czech Science Foundation under the project 23-05104S.

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.

Barták, R.; Svancara, J.; Skopková, V.; Nohejl, D.; and Krasicenko, I. 2019. Multi-agent path finding on real robots. *AI Commun.*, 32(3): 175–189.

Hönig, W.; Kumar, T. K. S.; Cohen, L.; Ma, H.; Xu, H.; Ayanian, N.; and Koenig, S. 2016. Multi-Agent Path Finding with Kinematic Constraints. In Coles, A. J.; Coles, A.; Edelkamp, S.; Magazzeni, D.; and Sanner, S., eds., *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling, ICAPS 2016, London, UK, June 12-17, 2016*, 477–485. AAAI Press.

Karpas, E.; Shleyfman, A.; and Tennenholtz, M. 2017. Automated Verification of Social Law Robustness in STRIPS. *Proceedings of the International Conference on Automated Planning and Scheduling*, 27(1): 163–171.

Nekvinda, M.; and Barták, R. 2021. Contingent Planning for Robust Multi-Agent Path Finding. In *33rd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2021, Washington, DC, USA, November 1-3, 2021*, 487–492. IEEE.

Paul, A.; Feng, Y.; and Li, J. 2023. A Fast Rescheduling Algorithm for Real-Time Multi-Robot Coordination [Extended Abstract]. *Sixteenth International Symposium on Combinatorial Search*, 16(1): 175–175.

Ratner, D.; and Warmuth, M. K. 1990. NxN Puzzle and Related Relocation Problem. *J. Symb. Comput.*, 10(2): 111–138.

Stergiou, K.; and Koubarakis, M. 2000. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence*, 120(1): 81–117.

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 Surynek, P.; and Yeoh, W., eds., *Proceedings of the Twelfth International Symposium on Combinatorial Search, SOCS 2019, Napa, California, 16-17 July 2019*, 151–159. AAAI Press.

Surynek, P. 2015. On the Complexity of Optimal Parallel Cooperative Path-Finding. *Fundam. Inform.*, 137(4): 517–548.

Svancara, J.; Obermeier, P.; Husár, M.; Barták, R.; and Schaub, T. 2023. Multi-Agent Pathfinding on Large Maps Using Graph Pruning: This Way or That Way? In Rocha, A. P.; Steels, L.; and van den Herik, H. J., eds., *Proceedings of the 15th International Conference on Agents and Artificial Intelligence, ICAART 2023, Volume 1, Lisbon, Portugal, February 22-24, 2023*, 199–206. SCITEPRESS.

Wurman, P. R.; D'Andrea, R.; Mountz, M.; and Mountz, M. 2008. Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses. *AI Magazine*, 29(1): 9–20.

Yu, J.; and LaValle, S. M. 2013. Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*.