Flexible, Lifelong, Explainable, and Robust Solutions for Multi-Agent Path Finding Problems

Aysu Bogatarkan

Sabanci University, Faculty of Engineering and Natural Sciences, Istanbul, Türkiye
aysubogatarkan@sabanciuniv.edu

Abstract

The multi-agent path finding (MAPF) problem is a combinatorial search problem that aims at finding paths for multiple agents in an environment without collisions, subject to constraints on the lengths of paths. The real-world applications of MAPF require flexible, lifelong, robust and explainable solutions. In this study, these challenges are being addressed.

For the success of Artificial Intelligence (AI) applications, two of the important features needed to be addressed by them are flexibility and explainability. A flexible AI method developed to solve a problem can accommodate variations of the problem, and an explainable AI method can provide answers to queries about the (in)feasibility and the optimality of solutions. Furthermore, it is desired for AI methods to provide robust and lifelong solutions for the problems. A robust solution for an AI application can still be used even after unexpected errors occur, and a lifelong AI method can adapt to changes during operation. One of the well-studied problems in AI that necessitates solutions for these challenges is the multi-agent path finding (MAPF) problem.

MAPF problem is a combinatorial search problem that aims to find paths for multiple agents in an environment such that no two agents collide with each other or obstacles, and subject to some constraints on the plan length. MAPF with constraints on plan lengths is intractable (Ratner and Warmuth 1986). Optimal solutions for MAPF are usually found by optimizing the makespan or the total plan length. MAPF has been studied in various domains, such as autonomous warehouse systems (Wurman, D'Andrea, and Mountz 2008).

This study focuses on introducing flexible, lifelong and robust methods for MAPF and its variants, and explainable frameworks for some MAPF variants. In all of our solutions, we utilize Answer Set Programming (ASP) (Lifschitz 2008)—a logic programming paradigm based on answer sets (Gelfond and Lifschitz 1988) and implement our methods using the ASP solver *clingo* (Gebser et al. 2011).

Flexible Solutions for MAPF

For some real-world applications, being able to solve MAPF problem may not be enough to address all challenges or the realistic conditions of the application. For instance, in real-world automated warehouses, the robots' battery levels change as they travel and they may need to be charged to complete their tasks. Furthermore, some parts of the warehouses with human occupants or tight passages may require robots to move slowly to ensure safety. One aim of our study is to address these issues with with flexible frameworks in the spirit of elaboration tolerance (McCarthy 1998).

To be able to address more realistic scenarios, a mathematical model general enough to handle multi-modal transportation conditions and multi-objective optimizations is needed. Furthermore, the computational framework is required to be flexible such that a large set of variations of MAPF problems can be addressed. Motivated by these challenges, we mathematically modelled a general version of MAPF (called mMAPF–multi-modal MAPF with resources) as a rich graph problem and introduced a flexible method to solve mMAPF declaratively, using ASP. Our method can handle the following variations of MAPF: *multi-objective optimization*, *waypoints*, *resource constraints* and *multi-modal transportation*. Details of this study can be found in our paper (Bogatarkan et al. 2020).

Explainable Solutions for MAPF

We also investigate the challenge of explainability for mMAPF problems, considering queries about the (in)feasibility and the optimality of solutions, along with queries about observations about these solutions. Given a solution for mMAPF, our explainable framework is able to explain infeasibility or nonoptimality of this solution, confirm its feasibility and suggest alternatives for the solution, and provide explanations for some queries, utilizing counterfactual reasoning and identifying violations of constraints.

For instance, suppose that an engineer would like to check whether some modifications of an mMAPF solution would be feasible or not. While an explanation regarding infeasibility could be "due to collisions with obstacles or other robots", an explanation regarding nonoptimality of a modification could be "because some more charging is required".

Alternatively, suppose that the engineer would like to better understand a given solution, and asks various queries about it. Queries and explanations would help the engineer to better understand the strengths and weaknesses of the solution, as well as the limitations of the infrastructure.

We introduced a method that considers different types of queries about mMAPF, and generates knowledge-rich explanations for each type of queries. For queries with affirmative answers, it generates alternative solutions as suggestions. For queries with negative answers, it utilizes counterfactual reasoning and weighted weak constraints to generate causality-based explanations and further recommendations. Since our method is query-based, utilizing the elaboration tolerance of ASP, it allows a sequence of interactive query answering by means of hypothetical reasoning.

Details of this study can be found in our paper (Bogatarkan and Erdem 2020).

Lifelong Solutions for MAPF

In a warehouse that is not completely autonomous, some changes may occur during the execution of a plan: existing agents may leave the environment, or new agents may be included in the team with new tasks, existing obstacles may be removed from the environment or moved to some other location in the environment. To be able to handle these changes, we have defined a general Dynamic MAPF (D-MAPF) problem and introduced multiple methods to solve this problem.

One of the possible solutions for D-MAPF is replanning: consider a new MAPF instance defined by the current locations and goal locations of both the existing and the new agents, and the updated environment, and compute a solution for this instance. Although replanning finds a solution, if one exists, it does not re-use the plans of the existing agents and may not be computationally efficient.

With this motivation, we proposed a novel method to solve D-MAPF, using ASP. The main idea is, instead of replanning for all the agents right away, to *revise and augment* the existing MAPF solution: (*revise*) try to schedule the waiting times of existing agents, (*augment*) while computing paths for the new agents. In this way, the paths for the existing agents can be re-used as part of the new plan.

In a more recent study, we investigated D-MAPF problem further. We introduced a rigorous definition for D-MAPF, that is general enough to cover 1) various changes in the environment and the team of agents over time, 2) different objective functions on plans, and 3) different assumptions on appearances/disappearances of agents, and that is not specifically oriented towards a particular method. We introduced a new framework to solve D-MAPF, that is general and flexible enough to allow different replanning and/or repairing methods. With the motivation of a modular architecture and efficient computations, our framework utilizes multi-shot computation (Gebser et al. 2019) of ASP.

To combine the advantages of multi-shot computation and re-using existing plans, we introduced a new method, called *Revise-and-Augment-in-Tunnels*. This method creates a "tunnel" for each existing agent, that consists of the agent's existing path and the neighboring locations within a specified "width". It revises the plans of existing agents within their own tunnels, while computing plans for the new agents at the same time, respecting the collision constraints.

We implemented all of our methods using multi-shot ASP, and integrated them in our D-MAPF framework. We designed and performed experiments to better understand the strengths and the weaknesses of our methods considering computational performance and quality of solutions.

Details of our studies on D-MAPF can be found in our papers (Bogatarkan, Patoglu, and Erdem 2019; Bogatarkan and Erdem 2025).

Ongoing and Future Work

Currently, we are working on the theoretical analysis for our methods by considering computational complexity and investigating correctness of our methods. Additionally, we have progressed in a novel method for defining robustness of MAPF plans, to be able to address more challenges of real-life applications. We are currently evaluating our method with experiments. In addition to new methods and problems, we are considering some real-life applications of MAPF and plan to demonstrate our methods on some selected real-world applications. For this purpose, we are collaborating with a logistics company.

References

Bogatarkan, A.; and Erdem, E. 2020. Explanation Generation for Multi-Modal Multi-Agent Path Finding with Optimal Resource Utilization using Answer Set Programming. *Theory Pract. Log. Program.*, 20(6): 974–989.

Bogatarkan, A.; and Erdem, E. 2025. A General Framework for Dynamic MAPF Using Multi-Shot ASP and Tunnels. *Theory and Practice of Logic Programming*, 1–18.

Bogatarkan, A.; Erdem, E.; Kleiner, A.; and Patoglu, V. 2020. Multi-modal Multi-agent Path Finding with Optimal Resource Utilization. In *Proceedings of 5th International Conference on the Industry 4.0 Model for Advanced Manufacturing*, 313–324.

Bogatarkan, A.; Patoglu, V.; and Erdem, E. 2019. A Declarative Method for Dynamic Multi-Agent Path Finding. In *Proc. of the Global Conference on Artificial Intelligence*, 54–67.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2019. Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming*, 19(1): 27–82.

Gebser, M.; Kaufmann, B.; Kaminski, R.; Ostrowski, M.; Schaub, T.; and Schneider, M. 2011. Potassco: The Potsdam Answer Set Solving Collection. *AI Commun.*, 24(2): 107–124.

Gelfond, M.; and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proceedings of International Logic Programming Conference and Symposium*, 1070–1080.

Lifschitz, V. 2008. What is Answer Set Programming? In *Proc. of AAAI*, 1594–1597.

McCarthy, J. 1998. Elaboration Tolerance. In *Proc. of CommonSense*.

Ratner, D.; and Warmuth, M. K. 1986. Finding a Shortest Solution for the N $\times$ N Extension of the 15-PUZZLE Is Intractable. In *Proc. of AAAI*, 168–172.

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