Multi-Agent Open Framework: Developing a Holistic System to Solve MAPF (Student Abstract)

Enrico Saccon

University of Trento
enrico.saccon@unitn.it

Abstract

Automation in industries is becoming an ever-increasing necessity, especially in the sector of logistics. In many cases, this means having many different automated guided vehicles (AGVs) moving at the same time, hence needing coordination to avoid conflicts between different agents. The problem of organizing a fleet of autonomous robots is known as the Multi-Agent Path Finding (MAPF) problem in the literature for which several optimal and sub-optimal algorithms have been proposed. When faced with real-life scenarios, these algorithms must provide the best feasible solution in the shortest time possible, therefore they must scale for large scenarios and be efficient. In this work, we briefly describe our open-source framework we are working on and we lay down the research paths we are going to focus on. The goal is to develop a holistic system that allows to control different aspects of the MAPF problem, from graph topology to goal scheduling.

Introduction

The Multi-Agent Path Finding (MAPF) problem (Standley 2010; Roni et al. 2019) is a well-known combinatorial problem that has been proven to be NP-hard (Yu and LaValle 2013), hence an optimal solution cannot be easily computed.

Optimal algorithms exist (Sharon et al. 2013, 2015; Surynek et al. 2016), but they tend to be slow and to scale poorly when faced with large environments, which is particularly important in scenarios where a new plan must be recomputed on the flight upon an unexpected event.

To better fit the scalability and efficiency necessities, sub-solvers are usually preferred to tackle the MAPF problem, providing either a bounded sub-optimal solution (Walker, Sturtevant, and Felner 2018), i.e., a solution that distances the optimal one by a maximum chosen margin, or an unbounded sub-optimal solution (Barer et al. 2014), i.e., a solution that may be much worse than the optimal, but is provided faster.

Anytime solvers provide a trade-off since they initially return an unbounded sub-optimal solution, and then, in the remaining time, they refine it by using Local Neighborhood Search (LNS) to isolate a sub-problem and solve it (Li et al. 2021; Huang et al. 2022).

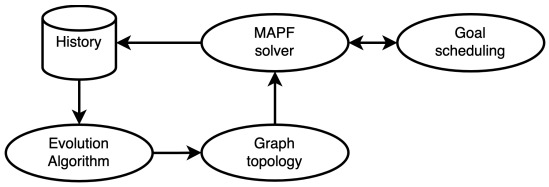

Figure 1: A diagram representing the system we aim at developing.

All these algorithms focus on solving the algorithmic part of the problem, but fail to take into consideration other aspects of the MAPF problem. Our goal is to create a system which does not simply solve the problem as is, but it may provide better support both on the short run and on the long run by considering also the topology of the network, different sets of goals and finally the interpolation and feasibility of the planned paths.

Problem Definition

We consider a modified MAPF problem, and a variant of the pick-up and delivery problem (Liu et al. 2019), in which each agent may have some intermediate goals that they must move through before reaching their final destination. The problem consists in moving k agents $\mathcal{A} = \{a_1, ..., a_k\}$ from their initial position s_i to their final position t_i through a series of intermediate goals $\mathbf{g}_i$ while minimizing a cost function, e.g., the sum of costs or the makespan.

Framework and Future Works

The studies mentioned in the introductory section are meant for single-goal reasoning, that is, they provide a solution for the agents moving from a source node to a target node. In real-life scenarios, this is usually not the case, but instead, multiple goals are assigned to the agents that they have to complete before moving to the target location.

Our main goal is to create a holistic system as shown in Figure 1, which would consider also other different aspects of the problem. In Figure 2, a search-based algorithm would move the two robots along the the shortest paths, ending up in a long series of swap conflicts increasing the search space. One of the two robots must use the longer road in order to solve the problem. There are two possibilities that we aim

at exploiting with our system. In a first instance, we could notice that the two goals are near the wrong agents, that is, if the agents were to reach the nearest "G" and then move to their target, then the problem would be much simpler to solve and it would lead to a far better solution.

Another possibility considers the fact that the agents cannot swap goals, e.g., if they had to load some goods too large or heavy for the other agents. Then, if this situation happens frequently, we could work on the graph topology by creating one-way corridors in order for the agent to not conflict with each other. To do this, a database of the past goals should be kept and an evolutionary algorithm may be run on it to verify whether some routes on the map are taken more often then others and if changes in the map topology may improve the results.

So far, we have developed a general open-source framework[1] written in C++ that aims at creating an environment in which it is possible to easily test different algorithms on different scenarios interchanging some aspects of the algorithms. We have written an implementation of three optimal solvers, namely CBS, ICTS and a MILP solver which uses CPLEX, and a rudimentary implementation of the anytime solver X* (Vedder and Biswas 2021). As for the single-agent solvers, we have coded A* and another solver based on Multi-Valued Decision Diagrams (MDDs) (Srinivasan et al. 1990). We are currently using an admissible heuristic for A* when solving X* conflicts that takes into consideration also the paths of the other agents in the sub-problem allowing to reduce the number of sub-conflicts to solve.

We also added the possibility of representing the map with different structures, either through an adjacency matrix, a grid or a graph. Indeed, the framework, and hence our problem, is not bound to unitary cost edges, but we could also model graphs with edges of different lengths. The possibility of having multiple choices on how to load the environment may lead to performance improvements in some scenarios, for example a grid may be better when there are a lot of nodes and fewer obstacles, whereas a graph may be better when there are more obstacles.

While at the moment the agents are considered as points moving on the map, future improvements will work on non-homogeneous agents and also consider the possibility that agents may need to recharge periodically.

One final aspects on which we are going to focus is motion planning. Indeed, once the MAPF algorithm has computed the best solution on the graph, the agents have to actually follow it. If the information on the kinematics constraint of the agents are not taken into consideration during the planning of the paths, then the resulting solution may be unfeasible to follow, leading to a waste of time and resources.

[1] https://gitlab.com/chaff800/MAOF

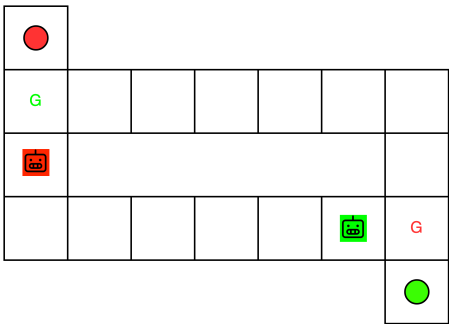

Figure 2: In this scenario, the robots should first move on the cell with the color-matched goal ("G") and then move to their final destination which is the corresponding circle.

References

Barer, M.; Sharon, G.; Stern, R.; and Felner, A. 2014. Suboptimal Variants of the Conflict-Based Search Algorithm for the Multi-Agent Pathfinding Problem.

Huang, T.; Li, J.; Koenig, S.; and Dilkina, B. 2022. Anytime Multi-Agent Path Finding via Machine Learning-Guided Large Neighborhood Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36: 9368–9376.

Li, J.; Chen, Z.; Harabor, D.; Stuckey, P. J.; and Koenig, S. 2021. Anytime Multi-Agent Path Finding via Large Neighborhood Search. 4127–4135. IJCAI.

Liu, M.; Ma, H.; Li, J.; and Koenig, S. 2019. Task and path planning for multi-agent pickup and delivery. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Roni, S.; Nathan, S.; Ariel, F.; Sven, K.; Hang, M.; Thayne, W.; Jiaoyang, L.; Dor, A.; Liron, C.; Satish, K. T. K.; Eli, B.; and Roman, B. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. *CoRR*, abs/1906.08291.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219: 40–66.

Sharon, G.; Stern, R.; Goldenberg, M.; and Felner, A. 2013. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence*, 195: 470–495.

Srinivasan, A.; Ham, T.; Malik, S.; and Brayton, R. 1990. Algorithms for discrete function manipulation. In *1990 IEEE Int. Conf. on Computer-Aided Design*, 92–95.

Standley, T. 2010. Finding optimal solutions to cooperative pathfinding problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 173–178.

Surynek, P.; Felner, A.; Stern, R.; and Boyarski, E. 2016. Efficient SAT approach to multi-agent path finding under the sum of costs objective. In *22nd ECAI*, 810–818.

Vedder, K.; and Biswas, J. 2021. X*: Anytime multi-agent path finding for sparse domains using window-based iterative repairs. *Artificial Intelligence*, 291: 103417.

Walker, T. T.; Sturtevant, N. R.; and Felner, A. 2018. Extended Increasing Cost Tree Search for Non-Unit Cost Domains. 534–540. IJCAI.

Yu, J.; and LaValle, S. M. 2013. Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs. 1443–1449. AAAI Press.