Improving the Sum-of-Cost Computation for Multi-Agent Pathfinding in Answer Set Programming

Roland Kaminski[1], Klaus Strauch[1] and Jiří Švancara[2]

[1] University of Potsdam, Germany
[2] Charles University, Czech Republic

1 Introduction

Multi-Agent Pathfinding [9, MAPF] is the problem of finding paths for agents in graphs while avoiding collisions. Agents may traverse a graph's edges or wait at vertices. All agents have a start vertex and must arrive at their goal vertex. Furthermore, agents must not occupy the same vertex at the same time and must not traverse the same edge in different directions at the same time.

Deciding whether a MAPF problem is satisfiable is a polynomial problem [6] and there are efficient algorithms to find solutions [8]. However, putting bounds on the number of moves of agents turns the decision problem into an NP-complete problem [5]. This makes the task of finding optimal solutions for MAPF problems much harder than finding an arbitrary solution. In this paper, we focus on finding optimal solutions subject to two different cost functions. The first one optimizes the length of the longest path among all agents (makespan) while the second one optimizes the sum of the path lengths of all agents (sum-of-cost).

Different types of solvers for MAPF are available. Among them, there are graph traversal-based algorithms such as Conflict-Based Search [2, CBS] or reduction-based methods that, for example, encode MAPF problems as SAT formulas [10]. Here, we focus on a reduction-based method using Answer Set Programming [3, 7, ASP] to encode MAPF problems and their cost functions as logic programs. We propose an ASP-based system using an improved algorithm to calculate sum-of-cost optimal solutions and compare it to existing systems.

2 Solving MAPF Optimally

Reduction-based solvers, to find makespan optimal solutions, follow a straightforward iterative deepening-based approach [10]. Given a MAPF problem and some initial value for a bound on the maximum path length of each agent, we increment the bound until the MAPF problem becomes satisfiable. We adapt this approach to our ASP-based system by encoding the bounded MAPF problem as a logic program to obtain solutions corresponding to bounded plans.

Finding a sum-of-cost solution is more involved. There are two main aproaches [1]; the *jump* and the *iterative* method. The *jump* method starts by finding a makespan

optimal plan. Using results from [1], we infer an upper bound on the maximum path length for each agent; the plans of the bounded MAPF problem include at least one sum-of-cost optimal plan. At this point, we can directly use ASP and its inbuilt optimization facilities to obtain solutions corresponding to sum-of-cost optimal plans.

The *iterative* method starts by putting individual bounds on the path lengths of agents and an additional bound on the sum-of-cost. The bounds for the agents are initialized with their shortest path lengths and the bound for the sum-of-cost to the sum of the length of the shortest paths. The method then tries to find a plan satisfying the individual bounds of the agents and the bound on the sum-of-costs. If no such plan exists, it proceeds by incrementing all bounds by one until a bounded plan is found. The first plan obtained in this manner corresponds to a sum-of-cost optimal plan. As before, the bounded MAPF problem is solved by a reduction to ASP.

A very important pre-processing step is to calculate the vertex/time point pairs for agents from which the agent's goal is still reachable given the current bounds [10]. This allows us to reduce the size of the logic programs and speed up solving. It is important to note that this optimization is especially effective for the sum-of-cost objective because we can exploit the individual bounds of agents to reduce the set of reachable vertices. In general, there are fewer reachable positions under the sum-of-cost objective than under the makespan objective.

3 Refinements

In this section we consider a refinement to the jump method presented in the previous section. We begin by noting that the first step to compute a makespan optimal plan is unnecessarily hard. Since the purpose of this plan is just to get an upper bound on the path length, it does not matter whether it is makespan optimal or not. So, instead of finding a makespan optimal plan, we employ the iterative method disregarding the bound on the sum-of-costs. This also allows us to employ the reachability optimization to obtain an initial plan with less computational effort as compared to computing a makespan optimal plan.

Finally, since we are not looking for an optimal solution in this first step, we can also increase individual agent bounds by larger increments potentially reducing the amount of solver calls needed to find that first model.

4 Benchmarks

We run benchmarks on a set of instances stemming from [4] of types random, maze, room, and empty with sizes 32×32, 64×64 and 128×128. All experiments were run with a timeout of 300s and a memory limit of 28GB. The results show that the old *jump* method is fastest for the 32×32 instances, which follows the results of [1]. However, for the larger instances, we observe that the old *jump* method is slowest, while our new *jump+2* method is fastest, where the *+2* stands for the increment discussed at the end of Section 3.

References

[1] R. Barták and J. Svancara. "On SAT-Based Approaches for Multi-Agent Path Finding with the Sum-of-Costs Objective". In: *Proceedings of the Twelfth International Symposium on Combinatorial Search (SOCS'19)*. Ed. by P. Surynek and W. Yeoh. AAAI Press, 2019, pp. 10–17.

[2] E. Boyarski et al. "ICBS: Improved Conflict-Based Search Algorithm for Multi-Agent Pathfinding". In: *Proceedings of the Twenty-fourth International Joint Conference on Artificial Intelligence (IJCAI'15)*. Ed. by Q. Yang and M. Wooldridge. AAAI Press, 2015, pp. 740–746. URL: http://ijcai.org/proceedings/2015.

[3] M. Gebser et al. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2012. DOI: 10.1007/978-3-031-01561-8.

[4] M. Husár et al. "Reduction-based Solving of Multi-agent Pathfinding on Large Maps Using Graph Pruning". In: *Proceedings of the Twenty-first International Conference on Autonomous Agents and Multiagent Systems (AAMAS'22)*. Ed. by P. Faliszewski et al. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2022, pp. 624–632. DOI: 10.5555/3535850.3535921.

[5] J. Ju and S. LaValle. "Optimal Multirobot Path Planning on Graphs: Complete Algorithms and Effective Heuristics". In: *IEEE Transactions on Robotics* 32.5 (2016), pp. 1163–1177.

[6] D. Kornhauser, G. Miller, and P. Spirakis. "Coordinating Pebble Motion On Graphs, The Diameter Of Permutation Groups, And Applications". In: *25th Annual Symposium onFoundations of Computer Science, 1984*. 1984, pp. 241–250. DOI: 10.1109/SFCS.1984.715921.

[7] V. Lifschitz. *Answer Set Programming*. Springer-Verlag, 2019. DOI: 10.1007/978-3-030-24658-7.

[8] R. Luna and K. Bekris. "Push and Swap: Fast Cooperative Path-Finding with Completeness Guarantees". In: *Proceedings of the Twenty-second International Joint Conference on Artificial Intelligence (IJCAI'11)*. Ed. by T. Walsh. IJCAI/AAAI Press, 2011, pp. 294–300.

[9] R. Stern et al. "Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks". In: *Proceedings of the Twelfth International Symposium on Combinatorial Search (SOCS'19)*. Ed. by P. Surynek and W. Yeoh. AAAI Press, 2019, pp. 151–159.

[10] P. Surynek et al. "Efficient SAT Approach to Multi-Agent Path Finding Under the Sum of Costs Objective". In: *Proceedings of the Twenty-second European Conference on Artificial Intelligence (ECAI'16)*. Ed. by G. Kaminka et al. IOS Press, 2016, pp. 810–818.