Bringing Multi-agent Path Finding Closer to Reality

Doctoral Consortium

Jiří Švancara
Charles University
Prague, Czech Republic
Jiri.Svancara@mff.cuni.cz

ABSTRACT

Multi-agent path finding is the problem of navigating multiple agents, located in a graph, from their current locations to their goal locations in such a way that there are no collisions between the agents. The classical definition of the problem assumes that the set of agents is unchangeable, and that the distances in the graph are homogeneous.

We propose to add to the problem specification a set of new attributes to bring it closer to the real world. These attributes include varying distances, number of agents that can occupy an edge or node, and dynamic appearance of new agents.

KEYWORDS

Multi-agent Path Finding, Satisfiability, Search Algorithms

ACM Reference Format:
Jiří Švancara. 2018. Bringing Multi-agent Path Finding Closer to Reality. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 2 pages.

1 INTRODUCTION

Multi-agent Path Finding (MAPF) is the task to navigate a set of mobile agents from their current locations to their desired locations while avoiding collisions with other agents [6]. In recent years, this problem received much attention in theoretical computer science. Applications range from robotics, traffic optimization, and warehouse management, to computer games and more [5].

An abstraction where the environment is represented by an undirected graph is often used in literature [3]. However, this graph does not include lengths for the edges and therefore all of them are treated as unit-length edges. Furthermore, all of the agents share the same properties, specifically all of the agents have the same speed and dimensions. Another thing to mention is that the set of the agents is unchangeable.

In this paper we propose to add new attributes to the classical MAPF problem. The attributes mainly include varying lengths for the edges over which the agents move as well as allowing new agents to dynamically appear in the graph. We also briefly study the implications of such new attributes.

2 DEFINITION

Formally a classical *MAPF instance* can be written as a pair (G, A), where $G = (V, E)$ is a graph and A is a set of agents. Each agent $a_i \in A$ is associated with starting location $s_i \in V$ and desired goal location $g_i \in V$. This means that every agent is also a pair $a_i = (s_i, g_i)$.

The time is discretized and in each time step every agent can perform either a move action to a neighboring node or stay in its current location. Let π_i denote a plan for agent a_i, then $\pi_i(j)$ denotes a location where agent a_i is present at time step j. A valid *solution of MAPF problem* is a plan

$$\pi = \bigcup_{a_i \in A} \pi_i$$

such that the following constraints are satisfied:

(1) The plan for each agent is a valid path. I.e. if $\pi_i(j) = v$ and $\pi_i(j+1) = u$ then $(v, u) \in E$ or $\pi_i(j) = \pi_i(j+1)$.
(2) No two agents occupy one node at the same time. I.e. for all pairs of agents a_{i_1} and a_{i_2} at all time steps j it holds that $\pi_{i_1}(j) \neq \pi_{i_2}(j)$.
(3) No two agents occupy one edge at the same time. I.e. for all pairs of agents a_{i_1} and a_{i_2} at all time steps j it holds that $\pi_{i_1}(j) \neq \pi_{i_2}(j+1) \vee \pi_{i_1}(j+1) \neq \pi_{i_2}(j)$.

Note that this definition allows agents to move along a fully occupied cycle as long as it contains 3 or more nodes.

In addition to having a valid solution, it is often required to find an optimal solution in terms of some cost function. The two most often used functions are *Sum of Costs* (SOC) [5] and *Makespan* [11].

3 RELATED WORK

The algorithms that solve the MAPF problem can be in general divided into two categories – suboptimal and optimal.

Suboptimal algorithms are mainly defined by not guarantying that the found solution is optimal. The main advantage is that they are computationally fast compared to the optimal algorithms. These algorithms can work based on a set of rules that are used repeatedly until a solution is found [2] or by taking advantage of the graph structure [8].

Optimal algorithms on the other hand are guaranteed to find an optimal solution in terms of some cost function. Whether the cost function is SOC or makespan, this problem is computationally hard [9].

The two main approaches to solve MAPF optimally is to use a search algorithm [4, 7] that may use some heuristic function [12] or to reduce the problem to some other formalism such as a satisfiability problem [10].

4 DISTANCES AND OCCUPANCY

The first extension to consider is to add lengths to the edges of the graph G. Now formally the graph is a triplet $G = (V, E, w)$, where

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden. © 2018 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

w is a function $w : V \to \mathbb{N}^+$ that assigns each edge a positive number. This number represents the number of time steps it takes any agent to move along this edge. We shall denote this extension as a *weighted MAPF*. If the function w assigns a unit-weight to each edge, then it is equivalent to the original MAPF problem.

One can argue that this extension can be reduced to the original problem by simply splitting the non-unit edges into appropriate amount of new edges with new nodes in-between. However, these two approaches are not equal as once an agent enters an edge, it must go through the whole edge without stopping or turning around. If the long edge is split into several unit edges, the agent can stop at any time or even turn around and go back. This is not corresponding with real world as, for example, a car can not turn around on highway at any time it wishes. Our paper on weighted MAPF will be published on AAMAS 2018 [1].

A new issue that arises with weighted MAPF is how many agents can be present at one time on one edge. In the original definition it is forbidden for more than one agent to be present to forbid swapping of two neighboring agents (two agents using the same edge in the opposite direction). In the original MAPF it is impossible for more than one agent to go in the same direction over one edge, because there could not be more agents present at the node to use the same edge in time. However, this is possible in weighted MAPF. If we leave the *occupancy* of an edge as a unit, this can lead to very different solutions than if we allow more agents to share the edge.

5 ONLINE VERSION

Another extension is to consider that the environment (graph) is constant and unchangeable, but the set of agents can change in a way that new agents can appear at some time steps. Formally this change agent a_i into a triplet $a_i = (s_i, g_i, t_i)$, where t_i is a time step when agent a_i appears in node s_i. However, a_i is revealed to the solver only at the time step t_i, thus making the problem on-line. We shall denote this extension as *online MAPF*.

One can again argue that this extension can be reduced to the original one by changing the graph. We add a directed path of length t_i that the agent needs to traverse before entering the original graph. For further reference, we will call this reduction *offline MAPF planner*. This is not equivalent to the online MAPF because offline MAPF planner does know in advance all of the agents, which is not an on-line problem.

There are two things to consider with regard to what happens with the new agent when it enters the graph and what happens when it reaches its goal.

If the agent a_i appears exactly at the time step t_i in the node s_i, this can lead to an unavoidable collisions, as there may be some agent already present at node s_i. To solve this problem, we propose that the agent will appear in some meta location and is required to perform one move action to enter the graph at node s_i. It is also allowed to wait in the meta location for as long as required.

If the agent remains in the goal node, after reaching its destination, it is easy to find an example that makes the instance of online MAPF unsolvable, however, it would be solvable by an offline MAPF planner that knows in advance when and where are the new agents going to appear. To solve this problem, we expect the agents to disappear from the graph after reaching their goal locations.

If we consider the two properties of appearance and disappearance of agents, it can be proven that every online MAPF problem can be solved iff there are paths from s_i to g_i for every $a_i \in A$.

On the other hand it can also be shown that there can not be any online MAPF solver that will guarantee as good solution as offline MAPF solver can produce. This is due to the fact that we do not know where the new agents will appear.

This extension has been also studied and a paper is under review for IJCAI 2018.

CONCLUSION

The mentioned extensions of the MAPF problem are studied in the Ph.D. thesis as well as different approaches to solve them. Mainly we are focused on SAT reduction solvers and search based solvers. As it turns out, if we add different attributes to the original problem, some solvers become less efficient and others may surpass them.

REFERENCES

[1] Roman Barták, J. Švancara, and M. Vlk. 2018. A Scheduling-Based Approach to Multi-Agent Path Finding with Weighted and Capacitated Arcs. In *To appear in Proceedings of AAMAS 2018, Stockholm, Sweden, July 11-13, 2018.*

[2] Ryan Luna and Kostas E. Bekris. 2011. Push and Swap: Fast Cooperative Path-Finding with Completeness Guarantees. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, Toby Walsh (Ed.). IJCAI/AAAI, 294–300. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-059

[3] Malcolm Ross Kinsella Ryan. 2008. Exploiting Subgraph Structure in Multi-Robot Path Planning. *J. Artif. Intell. Res.* 31 (2008), 497–542. https://doi.org/10.1613/jair.2408

[4] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. 2012. Conflict-Based Search For Optimal Multi-Agent Path Finding. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, Jörg Hoffmann and Bart Selman (Eds.). AAAI Press. http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5062

[5] Guni Sharon, Roni Stern, Meir Goldenberg, and Ariel Felner. 2011. The Increasing Cost Tree Search for Optimal Multi-Agent Pathfinding. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, Toby Walsh (Ed.). IJCAI/AAAI, 662–667. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-117

[6] David Silver. 2005. Cooperative Pathfinding. In *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference, June 1-5, 2005, Marina del Rey, California, USA*, R. Michael Young and John E. Laird (Eds.). AAAI Press, 117–122.

[7] Trevor Scott Standley. 2010. Finding Optimal Solutions to Cooperative Pathfinding Problems. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, Maria Fox and David Poole (Eds.). AAAI Press. http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1926

[8] Pavel Surynek. 2009. A novel approach to path planning for multiple robots in bi-connected graphs. In *2009 IEEE International Conference on Robotics and Automation, ICRA 2009, Kobe, Japan, May 12-17, 2009.* IEEE, 3613–3619. https://doi.org/10.1109/ROBOT.2009.5152326

[9] Pavel Surynek. 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*, Maria Fox and David Poole (Eds.). AAAI Press. http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1768

[10] Pavel Surynek. 2012. On Propositional Encodings of Cooperative Path-Finding. In *IEEE 24th International Conference on Tools with Artificial Intelligence, ICTAI 2012, Athens, Greece, November 7-9, 2012.* IEEE Computer Society, 524–531. https://doi.org/10.1109/ICTAI.2012.77

[11] Pavel Surynek. 2014. Compact Representations of Cooperative Path-Finding as SAT Based on Matchings in Bipartite Graphs. In *26th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2014, Limassol, Cyprus, November 10-12, 2014.* IEEE Computer Society, 875–882. https://doi.org/10.1109/ICTAI.2014.134

[12] Jiri Svancara and Pavel Surynek. 2017. New Flow-based Heuristic for Search Algorithms Solving Multi-agent Path Finding. In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence, ICAART 2017, Volume 2, Porto, Portugal, February 24-26, 2017.*, H. Jaap van den Herik, Ana Paula Rocha, and Joaquim Filipe (Eds.). SciTePress, 451–458. https://doi.org/10.5220/0006184504510458