Studying Online Multi-Agent Path Finding

Jonathan Morag, Roni Stern, Ariel Felner, Dor Atzmon, Eli Boyarski
Ben-Gurion University of the Negev
{moragj, sternron}@post.bgu.ac.il, felner@bgu.ac.il, {dorat, boyarske}@post.bgu.ac.il

Abstract

Multi-agent path finding (MAPF) is the problem of planning a set of non-conflicting plans on a graph, for a set of agents. Online MAPF extends MAPF by considering a more realistic problem in which new agents may appear over time. While planning, an online solver does not know whether and which agents will join in the future. Therefore, in online problems the notion of snapshot-optimal was defined, where only current knowledge is considered. The quality of such a solution may be weaker than the quality of a solution to an equivalent offline MAPF problem (offline-optimality), where the solver is preinformed of all the agents that will appear in the future. In this paper we explore, theoretically and empirically, the quality of snapshot-optimal solutions compared to offline-optimal solutions.

Definitions and Background

The input to the *Multi-Agent Path Finding* (MAPF) problem is a tuple $\langle G, A\rangle$, where $G = (V, E)$ is a graph, and $A = \{a_1, ..., a_k\}$ is a set of k agents. Each agent a_i is associated with a start vertex $s_i \in V$ and a goal vertex $g_i \in V$ (Stern et al. 2019). A solution to a MAPF problem is a list $\pi = \{\pi_1, \dots, \pi_k\}$ of individual agent plans such that each agent $a_i \in A$ is associated with a single plan π from its start to its goal. π is a sequence of vertices such that $\pi_i(t) \in V$ is the planned vertex for agent a_i at time t, where time is discrete. $\pi_i[x] \in V$ (note the difference between parenthesis () and brackets []) is the vertex that the agent would reach after performing x moves according to its plan. Thus, $\pi_i[0] = s_i$ and $\pi_i[|\pi_i| - 1] = g_i$. At each time step an agent can either move or wait at its current vertex. The length of a plan is defined as ($len(\pi_i) = |\pi_i| - 1$). A MAPF solution is *valid* only if none of the plans within it conflict. Two plans π_i and π_j conflict if at any time t the two agents are planned to occupy the same vertex, or are planned to swap their vertices (Stern et al. 2019). A common cost function in MAPF is the Sum Of Costs (SOC), defined as the sum of the lengths of all individual plans, $SOC(\pi) = \sum_{i=1}^{k} len(\pi_i)$. A MAPF solution is *optimal* if it is valid and has the minimal cost among all valid solutions.

Most previous works on MAPF focused on the *offline MAPF* setting, where plans are found before the agents start their movement, and are assumed to execute them without any modification during execution. Recently, an *online MAPF* setting was suggested (Švancara et al. 2019). In this setting, new agents may appear over time and wish to join the problem space while existing agents are still executing their plans. Appearing agents are allowed to wait outside the problem space before entering and when agents reach their goals, they disappear. This problem is relevant to real-world problems, such as autonomous intersections, robot warehouses, airport traffic etc.

In online MAPF, each agent a_i is associated with a time of appearance $TOA(a_i)$. Accordingly, each plan π_i in an online MAPF solution starts at that agent's time of arrival ($\pi_i[0] = \pi_i(TOA(a_i))$). Note that the new agents are only revealed to an online solver at the time of their arrival.

The solution to online MAPF is a sequence of all the solutions found at the different times when new agents appeared $\Pi = \{\pi^0, ..., \pi^n\}$, where π^t is the solution found at time t. The executed solution $Ex(\Pi)$ represents the plans that the agents ended up following, considering the changes made to their plans over time. An online MAPF solution Π is valid only if none of the plans in $Ex(\Pi)$ have a conflict. The cost of an online solution Π is $SOC(Ex(\Pi))$.

Any online MAPF problem may be converted into an equivalent offline problem by informing the solver of all the agents that will appear in the future. By optimally solving the equivalent offline problem, we can find the lowest cost possible for a solution to the online problem. We designate such a solution as *offline-optimal* (or oracle optimal) solution. Naturally, online problems can not be solved offline in practice. However, the cost of the offline-optimal solution is useful for theoretical comparisons.

An online solver for online MAPF does not know when and which agents may appear in the future. Consequently, a solution with a lower cost than that of the executed solution may exist. No online MAPF solver can guarantee to find a solution with cost equal to the offline-optimal cost of an equivalent offline problem (Švancara et al. 2019). An algorithms that always returns a solution for the current set of agents that is optimal, assuming no new agents arrive in the future, is called *snapshot-optimal* (Švancara et al. 2019).

Snapshot-Optimal Vs. Offline-Optimal

In this section, we explore how optimal snapshot-optimal is, by comparing it with offline-optimal.

Theoretical Comparison

Let P be an online MAPF problem. Let $\Pi_{oo}(P)$ and $\Pi_{so}(P)$ be the solutions to P generated by an offline-optimal solver and snapshot-optimal solver, respectively. Let P^+ be an online MAPF problem that is identical to P except that there is an additional new agent a_i and $\forall a_j \neq a_i(TOA(a_i) > TOA(a_j))$. Let $t^+ := TOA(a_i)$. Let π_i^* be a plan for agent a_i that has minimal length while ignoring all other agents. Let $\Delta_{SOC}(P)$ be the difference in SOC between $\Pi_{oo}(P)$ and $\Pi_{so}(P)$, that is,

$$\Delta_{SOC}(P) = SOC(\Pi_{so}(P)) - SOC(\Pi_{oo}(P)).$$

Note that if $\Delta_{SOC}(P^+) > \Delta_{SOC}(P)$ it means that adding a new agent caused the cost difference to increase.

Observation 1. If $\Delta_{SOC}(P^+) > \Delta_{SOC}(P)$ then it must hold that for every minimal length plan for the new agent π_i^* there exists a conflict with the plans of the other agents in any $\Pi_{so}(P)$.

Observation 2. If $\Delta_{SOC}(P^+) > \Delta_{SOC}(P)$ then it must hold that for every snapshot-optimal solution $\Pi_{so}(P^+)$ either there exists an old agent whose plan is longer than its plan in $\Pi_{so}(P)$, or $len(\Pi_{so}(P^+)_i) > len(\pi_i^*)$.

A solution π is applicable at time step t for an online solution Π if $\forall i(Ex(\Pi)_i(t) = \pi_i(t))$, meaning all agents occupy the same vertices in both solutions at t (including agents that already disappeared) A solution Π' is a prefix of Π up to time step t if it is applicable at every time step $t' < t$ for Π.

Observation 3. If $\Delta_{SOC}(P^+) > \Delta_{SOC}(P)$ then for every offline-optimal solution $\Pi_{oo}(P^+)$ and snapshot-optimal solution $\Pi_{so}(P)$, it holds that $\Pi_{so}(P)$ is not a prefix of $\Pi_{oo}(P^+)$ up to time step t^+.

From these observations we conclude that for a significant difference in quality to exist between an offline-optimal solution and a snapshot-optimal solution for an online MAPF problem, that problem would have to contain many situations where scenarios implied by the observations are met.

Experimental Comparison

We used problem instances from a common MAPF benchmark (Stern et al. 2019). The benchmark contains a set of grid-maps (often seen as 4-connected), and a set of offline MAPF scenarios for each map. We made the following adjustments to the instances to create online MAPF instances:

1. For each online scenario a set of 20 agents was selected randomly from an existing offline scenario.

2. For every time step, the number of new agents that appear was drawn from a Poisson distribution. The rate of agent appearance is set by the distribution's λ parameter.

3. Start-goal pairs for agents were drawn from a standard normal distribution over the set of pairs. We chose this method as we felt it would reflect real-world scenarios.

We adapted the MAPF solver CBS (Sharon et al. 2015) to compare the quality of offline-optimal and snapshot-optimal solutions on 50 online MAPF instances per map and appearance rate. The appearance rates (λ) ranged from 0.05 to 1.5 agent per time step, with 40 agents per instance. We used two 32x32 grid maps: 'random-32-32-20', which has 20% random obstacles, and 'room-32-32-4' which has randomly connected 4x4 rooms (Figure 1). Each solver was allowed 300 seconds to solve each instance. For the instances solved by both solvers and on all arrival rates, we measured the average cost of offline-optimal and snapshot-optimal solutions and the proportion of instances where no cost difference was observed. The results of this experiment are found in Table 1. We see that the cost of snapshot-optimal solutions was on average very close to the offline-optimal cost, and that in most problem instances they were equal.

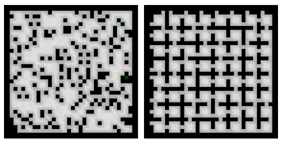
Figure 1: random-32-32-20 (left), room-32-32-4 (right)

Map	Rate	Offline-Optimal	Snapshot-Optimal	% Equal
random-32-32-20	0.05	856.34	856.43	98.11%
	0.50	859.51	859.79	91.51%
	1.00	862.68	863.28	88.68%
	1.50	866.34	867.04	90.57%
room-32-32-4	0.05	915.04	915.32	96.00%
	0.50	920.12	921.28	82.00%
	1.00	921.60	922.72	86.00%
	1.50	925.96	926.04	98.00%

Table 1: Comparison of snapshot and offline optimal solutions for 40 agents, with varying appearance rates.

The experiment results confirm our claim that the scenarios implied by the observations above are rare. Therefore, we conclude that snapshot optimal algorithms are very powerful as they are practical to implement and their solution quality is very close to that of an offline (oracle) optimal solution.

Online MAPF Variants and Conclusions

We also developed and analyzed lifelong versions of snapshot-optimal solvers, concluding that in most cases the improvement of such versions is only marginal. We further evaluated a setting where rerouting an agent carries a specific cost. We found that in the tested scenarios, solving sub-optimally while completely avoiding reroutes produced higher quality solutions under most reroute costs. Finally, we examined a setting where agents are assigned different priorities, and showed a trade-off between solution quality and problem coverage in this setting.

In conclusion, we examined the matter of optimality in online Multi-agent Path Finding. We defined offline-optimality in the online MAPF problem, and compared it to snapshot-optimality both analytically and empirically. We conclude that snapshot-optimal solutions are usually very similar in quality to offline-optimal solutions.

References

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

Stern, R.; Sturtevant, N. R.; Felner, A.; Koenig, S.; Ma, H.; Walker, T. T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. K. S.; Boyarski, E.; and Bartak, R. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. *Symposium on Combinatorial Search (SoCS)* 151–158.

Švancara, J.; Vlk, M.; Stern, R.; Atzmon, D.; and Barták, R. 2019. Online multi-agent pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7732–7739.