An Online Approach for Multi-Agent Path Finding Under Movement Uncertainty (Extended Abstract)

Elad Levy, Guy Shani, Roni Stern,

Software and Information Systems Engineering, Ben Gurion University of the Negev, Israel
elad.levy2@gmail.com,shanigu@bgu.ac.il,sternron@bgu.ac.il

Introduction

The multi agent path finding (MAPF) problem deals with a group of agents in a discrete environment, where each agent aims to reach a given goal location without colliding with the other agents (Stern et al. 2019). MAPF has many real world applications, such as robots operating in a logistics center (Wurman, D'Andrea, and Mountz 2008; Salzman and Stern 2020), airport towing (Morris et al. 2016), autonomous vehicles, robotics (Veloso et al. 2015), and digital entertainment (Ma et al. 2017). Classical MAPF research deals focused on deterministic domains, where there is only a single possible effect for each action. However, real-world problems, e.g., in robotic scenarios, often exhibit stochastic behaviors. For example, the wheels of a robot attempting to move forward may slip, causing it to move sideways, or remain in place. Often, one can define a probability distribution over the possible outcomes of an action, defining a stochastic problem that can be modeled as a Markov decision process (MDP). Previous research on MAPF has considered stochastic settings, but limited to cases where the agent can either move to its intended direction or remain in place (Wagner and Choset 2017; Atzmon et al. 2020). In such stochastic settings, the set of locations that an agent may reach when executing a plan still forms a path in the environment. Hence, classical MAPF algorithms that compute a sequence of actions may still provide a viable solution in these cases. Indeed, most existing work on stochastic MAPF build on algorithms for classical (deterministic) MAPF.

We focus on a more general type of stochastic MAPF problem, where agents move in directions that differ from the intended direction with some probability. This seemingly minor generalization makes the problem considerably harder, as a solution now must specify a *policy* for each agent, i.e., a mapping from any position the agent may reach to an appropriate action. As a result, standard approaches to solve classical MAPF such as A*-based solvers (Standley 2010; Wagner and Choset 2011, 2017) or CBS (Sharon et al. 2015) do not transfer easily. Our contributions include formalizing the problem of stochastic MAPF, proposing an online approach to solve it, and comparing it experimentally to offline MDP-based solutions.

Stochastic MAPF

A stochastic MAPF problem is a tuple $\langle G, n, S, T, P, C\rangle$ where $G = (V, E)$ is a graph of possible locations; n is the number of agents; $S = (s_0, \dots, s_n)$ is the initial locations; $T = (t_0, \dots, t_n)$ is the target locations; $P(v, a, v')$ is the single-agent stochastic transition function specifying the probability that an agent reaches v' when doing action a when at state v; and $C(v, a)$ is the single-agent cost function specifying the cost of applying action a at position v. The joint state of the system $\mathbf{v} = \langle v_1, ..., v_n\rangle$ comprises the current location of each agent. The joint action $\mathbf{a} = \langle a_1, ..., a_n\rangle$ specifies an action for each agent. Transitions and costs are independent, i.e., the probability of reaching a joint state **v'** when doing a joint action **a** at state **v** is the product of its constituent single-agent transition probabilities and the cost of a joint action is the sum of its constituent single-agent costs. Collisions are strictly forbidden, that is, a joint action **a** cannot be executed at a state **v** if there is a non-zero probability to reach a state $\langle v'_1, ..., v'_n\rangle$ where there exist $i \neq j$, such that $v_i = v_j$. A solution to a stochastic MAPF problem is a joint policy assigning for each joint state a joint action. A policy is valid if it never assigns a forbidden joint action to any joint state. A policy is optimal if it minimizes the expected sum of costs for all agents reaching their goals.

A stochastic MAPF problem can be modeled as an MDP over the joint state and action spaces. Therefore, it can be solved by any MDP solution method, such as Value Iteration (VI) and Real-Time Dynamic Programming (RTDP) (Barto, Bradtke, and Singh 1995). While this algorithms work well for the single-agent case, they struggle when faced with the large search space and branching of stochastic MAPF. In practice, even scaling to 3 agents often proves to be prohibitively difficult.

Online Conflict Detection and Resolution

We propose the following online approach to solve the stochastic MAPF problem. Initially, we compute *offline* a single agent policy for each agent independently, ignoring potential collisions. The agents then begin to execute their policies until either all agents reached their goals or some agents come within a predefined distance of each other and their current policies may lead to a collision.

Once a potential collision between a set of agents was detected, we resolve it locally, as follows. First, we identify the

Method	N	$E[C]$	T_{of}	T_{on}	S	R
Random-64-64-10						
Offline	2	198.3 ± 2.4	116.0	< 0.1	25	100.0
Online	2	199.8 ± 2.2	10.3	< 0.1	25	100.0
Offline	3	296.1 ± 5.2	180.0	1.8	25	77.8
Online	3	289.0 ± 4.9	15.3	0.1	25	100.0
Room-32-32-4						
Offline	2	84.1 ± 3.5	21.0	< 0.1	25	86.9
Online	2	85.1 ± 3.3	1.2	< 0.1	25	100.0
Offline	3	120.9 ± 6.3	180	0.3	6	62.0
Online	3	124.6 ± 3.8	1.7	0.1	6	100.0

Table 1: Results on 64x64 and 32x32 grids, comparing offline RTDP and our online method.

N	$E[C]$	T_{of}	T_{on}	MG	CR	S
Random-64-64-10						
4	382.6 ± 5.7	14.6	0.1	2	4.6	22
6	608.6 ± 8.0	22.0	0.2	2	9.0	16
8	741.6 ± 10.7	26.7	0.3	2	2.9	7
10	730.0 ± 8.6	35.7	0.4	2	8.2	3
ost003d						
4	754.7 ± 7.8	569.4	0.1	2	4.0	25
6	1185.0 ± 9.6	905.1	0.3	3	6.2	23
8	1605.8 ± 11.2	1215.1	0.4	2	9.2	20
10	2039.9 ± 13.1	1509.6	0.6	2	10.0	13
12	2381.3 ± 12.8	1771.4	2.4	3	15.5	4

Table 2: Results for the online methods on larger problems.

conflict area, which is the minimal region that contains all the possibly colliding agents. Then, we define a joint MDP including only the conflicting agent over the conflict area. The reward function for this joint MDP assigns reward to joint states in which the agent exist the conflict area. To steer agents towards their goals, we provider higher reward the agents for exiting the area in the direction they originally intended to follow. Our algorithm solves this joint MDP and the agents follow the resulting policy until either they exit the conflict area or a new potential collision is detected. This may occur because another agent reached the vicinity of the conflict area. In that case, we define a new area, as above, and the process is repeated.

Empirical Evaluation

We conducted a set of experiments to compare offline methods, where a complete policy is computed prior to execution, and the online method we proposed. The offline methods we considered are VI and RTDP over the joint MDP described in Section . In our experiments VI was not able to solve any problems with more than a single agent. Thus, we only report on the results of the offline RTDP. RTDP was run until it guaranteed convergence to the optimal policy with an error of less than 0.1, or a timeout of 180 seconds was reached.

To conduct our experiments, we used the standard 4-neighborhood grid classical MAPF benchmark (Stern et al. 2019). This benchmark contains a set of maps (grids), and each map has 25 different scenarios, specifying the start and goal positions of the agents. In all the experiments below we use all 25 scenarios, and run each method 30 times. We report the amount of scenarios where a method was able to achieve the goal in all 30 runs. To introduce uncertainty we assumed that in most grid locations, the agent moves in the intended direction, stays in place, moves clockwise, or counter-clockwise, with probabilities 0.7, 0.7, 0.7, and 0.1, respectively. The other grid locations are different: the probability to stay in place is increased to 0.2 and the probability to move towards the intended direction is decreased accordingly to 0.6. We added these special locations in the middle of the shortest paths between start and goal locations of all agents, and set their size to to $min(l/8, 3)$, where l is the length of the shortest path that is being patched.

Table 1 shows results of the offline RTDP (denoted as "Offline") and our online method, on two types of maps: a 64x64 grid with 10% randomly blocked cells and a 32x32 grid structured as a grid of 4x4 rooms. The table columns show the number of agents (N), the estimated expected cost and its standard error ($E[C]$), the offline and online computation time (T_{of} and T_{on}), the number of scenarios solved out of the available 25 scenarios (S), and the portion of episodes where the agents reached the goals in these scenarios (R).

As expected, the offline RTDP failed to scale, and could not converge to a policy that always reaches the goal (R score lower than 100). In contrast, our online solves these problems easily. Note that in one case — 64x64 grid with two agents — both online and offline solvers found a policy that always reaches the goal. Yet, the difference in expected cost was only slightly better for the offline RTDP. This suggests that while our online method is not provably optimal, it yields efficient policies.

Table 2 shows the results of our online method larger problems, where the offline method could not solve any instance. Here we also included `ost003d`, a large 194×194 game grid map. Column S shows the number of scenarios our agent managed to reach the goal in at least 1 episode. Column MG is the maximal encountered group of agents in a conflict area and column CR is the average number of conflict resolutions. Our results show we are able to solve even some problems with 10 and 12 agents, suggesting the potential scalability of our online method.

Conclusion and Future Work

We suggested an online approach for solving MAPF problems with stochastic effects. In our approach, conflicts are detected during execution (online) and resolved by a focused replanning of the conflicting agent over the conflicting region. We implemented this online approach and demonstrated its benefit over of an offline approach. There are much room for future work. First, our single agent policies are an interesting case where methods such as RTDP that focus on states visited using an optimal policy is a misfit. It would be interesting to develop faster methods for such problems, e.g., by identifying possible conflict areas and forcing RTDP to explore their vicinity. This would allow us to compute single agent policies faster.

Acknowledgements

This work has been partially funded by ISF grant #210/17 and NSF-BSF grant to Roni Stern.

References

Atzmon, D.; Stern, R.; Felner, A.; Sturtevant, N. R.; and Koenig, S. 2020. Probabilistic Robust Multi-Agent Path Finding. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 29–37.

Barto, A. G.; Bradtke, S. J.; and Singh, S. P. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1): 81–138.

Ma, H.; Yang, J.; Cohen, L.; Kumar, T. K. S.; and Koenig, S. 2017. Feasibility Study: Moving Non-Homogeneous Teams in Congested Video Game Environments. In *Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 270–272.

Morris, R.; Pasareanu, C. S.; Luckow, K. S.; Malik, W.; Ma, H.; Kumar, T. S.; and Koenig, S. 2016. Planning, Scheduling and Monitoring for Airport Surface Operations. In *AAAI Workshop: Planning for Hybrid Systems*.

Salzman, O.; and Stern, R. Z. 2020. Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems blue sky ideas track. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 1711–1715.

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

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

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.

Veloso, M. M.; Biswas, J.; Coltin, B.; and Rosenthal, S. 2015. CoBots: Robust Symbiotic Autonomous Mobile Service Robots. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Wagner, G.; and Choset, H. 2011. M*: A complete multirobot path planning algorithm with performance bounds. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3260–3267.

Wagner, G.; and Choset, H. 2017. Path Planning for Multiple Agents under Uncertainty. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 577–585.

Wurman, P. R.; D'Andrea, R.; and Mountz, M. 2008. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine*, 29(1): 9–9.