Proceedings of the Twelfth International
Symposium on Combinatorial Search (SoCS 2019)

Probabilistic Robust Multi-Agent Path Finding

Dor Atzmon, Ariel Felner, Roni Stern
Ben-Gurion University, Israel
dorat@post.bgu.ac.il, felner@bgu.ac.il, sternron@post.bgu.ac.il

A *Multi-Agent Path Finding* (MAPF) problem is defined by a graph $G = (V, E)$ and a set of agents $\{a_1 \ldots a_n\}$. At each time step, an agent can either *move* to an adjacent location or *wait* in its current location. The task is to find a plan π_i for each agent a_i that moves it from its start location $s_i \in V$ to its goal location $g_i \in V$ such that agents do not *conflict*, i.e., occupy the same location at the same time.[1]

In practice, unexpected events may delay some of the agents, preventing them from following the plan. Thus, it is desirable to generate a *robust plan* that can withstand such delays. Recently, a form of robustness called k*-robust MAPF* was introduced (Atzmon et al. 2018), in which each agent can be delayed up to k times and no collision will occur. In some cases, it is possible to estimate the probability that a delay will occur. In such cases, solving all conflicts with the same fixed value k may be less reasonable, and we might prefer solving conflicts based on their probabilities to occur. To this end, we explore a new form of robustness, p-robust, where a p-robust plan is a plan that can be executed without any collisions with a probability $\geq p$.

p-Robust CBS

pR-CBS is a CBS-based algorithm (Sharon et al. 2015) designed to return p-robust plans. To present pR-CBS, we introduce the notion of *potential conflict* and its relation to finding p-robust plans.

Definition 1 (Potential Conflict) *A plan π has a potential conflict $C = \langle a_i, a_j, t\rangle$ iff there exists $\Delta(C) \geq 0$ such that agents a_i and a_j are located in the same location in times t and $t+\Delta(C)$, respectively, i.e, when $\pi_i(t) = \pi_j(t+\Delta(C))$.*

A potential conflict $C = \langle a_i, a_j, t\rangle$ is said to *have occurred* if agent a_i experienced exactly $d_i \geq \Delta(C)$ delays before performing the t^{th} action in π_i, and agent a_j experienced exactly $d_i - \Delta(C)$ delays before performing the $t + \Delta(C)$ action in π_j. This means the agents will collide since $\pi_i(t) = \pi_j(t+\Delta(C))$ (they will collide at time $t+d_i$).

Let $P_0(\pi)$ be the probability that no potential conflict will occur when following plan π with a delay probability of p_d. It is easy to see that a plan π is p-robust iff the probability that no potential conflicts will occur is $\geq p$, i.e., $P_0(\pi) \geq p$.

pR-CBS is different than CBS in how it handles CT nodes, in how it chooses and resolves conflicts, and in how it orders nodes in the high-level open list.

Handling a CT node. When a CT node N is chosen for expansion, pR-CBS scans $N.\pi$ for potential conflicts by checking for locations occupied by more than one agent (even in different time steps). Then, $N.\pi$ is sent to a binary verifier that returns whether the plan is p-robust (if $P_0(\pi) \geq p$) or not. If the verifier returns TRUE, then the CT node is declared as goal and π is returned. If the verifier returns FALSE, it also returns $P_0(\pi)$ as well as a probability $P_{First}(C)$ for each potential conflict C. $P_{First}(C)$ provides the probability that while executing π, conflict C will occur first in time among all potential conflicts. Note that the sum of P_0 and all P_{First} probabilities equals 1, because either the execution succeeded (P_0) or one of the conflicts have occurred (one of the P_{First}).

Choosing a Conflict to Resolve. p-robust solution may contain potential conflicts. Thus, we need to resolve a set of conflicts such that the solution will be p-robust. pR-CBS chooses to resolve the conflict with the highest probability of occurring (highest P_{First}). This is a greedy approach that has high chances to reach a p-robust plan quickly, as it has the highest potential to increase P_0 in its children.

Resolving a Conflict. Let $C = \langle a_i, a_j, t\rangle$ be the chosen potential conflict in a non-goal node N. To resolve C, we add the range constraints $\langle a_i, \pi_i(t), [t, t + \Delta(C)]\rangle$ and $\langle a_j, \pi_j(t + \Delta(C)), [t, t + \Delta(C)]\rangle$ to a_i and a_j, respectively. This assures that these agents will not both be at the conflicting location in the time frame $[t, t + \Delta(C)]$.

Choosing CT Nodes. In this paper we focused on finding a p-robust plan as fast as possible. Therefore, we implemented a greedy approach that chooses to expand the node with highest P_0 value. Then, when a CT node with $P_0 \geq p$ is found by a verifier, that node is returned as a goal.

Statistical Verifiers

We describe two verifiers that verify statistically whether P_0 is *greater than or equal to* the desired robustness (p).

Fixed Verifier. The fixed verifier is first initialized with the following given parameters: p, p_d, s, and α, where p is the desired robustness, p_d is the constant delay probability,

[1]This research is supported by ISF grants no. 210/17 to Roni Stern and #844/17 to Ariel Felner and Eyal Shimony, by BSF grant #2017692 and by NSF grant #1815660

s is the number of simulations to be performed, and $1-\alpha$ is the confidence level of the statistical test. Then, a *critical value* c_1 is calculated by performing a Z-test as follows:

$$c_1 = p + Z_{1-\alpha} \cdot \sqrt{\frac{(1-p)\cdot p}{s}} \tag{1}$$

c_1 is calculated once and used later in every verification to determine whether $P_0 \geq p$ within the confidence level $1-\alpha$.

After pR-CBS has chosen to expand node N, it calls the fixed verifier to verify statistically whether N is a goal node. The verifier executes s simulations of the given plan ($N.\pi$) with delay probability p_d. To count collisions during executions we create a table (named $occurred$) that maps a given potential conflict to the number of times it has occurred. We also initialize a parameter: $successes$ that counts the number of executions in which no collision has occurred. During each execution, if a collision has occurred at a potential conflict C, the execution halts, and we increment $occurred[C]$ (initialized as 0). Otherwise, if no collision has occurred, we increment $successes$. When all s simulations ended, it sets $P_0 \leftarrow successes/s$. If $P_0 > c_1$, it returns TRUE. Otherwise, it sets $N.P_0 \leftarrow P_0$. For each conflict C it sets $N.P_{First}(C) \leftarrow occurred[C]/s$ and it returns FALSE.

Dynamic Verifier. This verifier chooses dynamically the number of simulations s to be performed in every CT node, as follows. First, it performs the minimum number of simulation that guarantees that $c_1 < 1$, which is derived from Equation 1 to be $\left\lceil Z_{1-\alpha}{}^2 \cdot \frac{p}{1-p} \right\rceil$. If $P_0 > c_1$ then we return TRUE. Otherwise, we might be able to perform more simulations until $P_0 > c_1$. However, the test might always fail and this will lead to an infinite loop. To overcome this issue, before executing more simulations, we perform another statistical test that checks whether $P_0 < c_2$ where c_2 is a new critical value which is calculated as follows:

$$c_2 = p - Z_{1-\alpha} \cdot \sqrt{\frac{(1-p)\cdot p}{s}} \tag{2}$$

If the second test passes, return FALSE. Otherwise, perform one more simulation, and check these tests again. The verification phase of the dynamic verifier summarized as follows. (1) Run s simulations and approximate P_0. (2) Calculate c_1 (Equation 1). (3) If $P_0 > c_1$, return TRUE. (4) Calculate c_2 (Equation 2). (5) If $P_0 < c_2$, return FALSE. (6) $s \leftarrow s+1$, run one more simulation, and goto step 2.

Experimental Results

We compared the performance of pR-CBS for different values of p with our two verifiers. In all of the following results $\alpha = 0.05$, $p_d = 0.2$, and P_0 was calculated based on 50 executions of the solution.

We compared standard CBS and pR-CBS with the fixed verifier for different values of p (0.7 and 0.9) and $s = 40$, on an 8x8 open grid with 8 randomly allocated agents. Table 1 presents the average cost, planning time (in ms), and P_0 for 60 problem instances. We can see that larger p increases the cost and time but results in less collisions (higher P_0). The optimal solver achieved the lowest cost (38.5) and the fastest planning time (only 9ms) with a tradeoff that many collisions occurred and only 41% of the executions were collision-free (P_0).

	Cost	Time(ms)	P_0
CBS	38.5	9	0.41
$p = 0.7$	43.3	7,620	0.84
$p = 0.9$	50.1	37,501	0.95

Table 1: Average planning cost, runtime and for CBS and pR-CBS with different values of p, over 8x8 open grid.

#Simulations	$p = 0.80$	$p = 0.85$	$p = 0.90$	$p = 0.95$
20	**59**	0	0	0
40	58	57	**57**	0
60	57	55	55	0
160	58	56	52	49
Dynamic	**59**	**59**	**57**	**52**

Table 2: Success rate for pR-CBS out of 60 instances.

We also compared the fixed verifier (with a different number of simulations) and the dynamic verifier, with $p = 0.8, 0.85, 0.9,$ and 0.95. 60 instances were generated and we present the number of instances that could be solved within 5 minutes in Table 2. As expected, if the number of simulations was too small, the fixed verifier could not solve any instance as a result of the statistic test (c_1 was greater than 1). On the other hand, the dynamic verifier could solve instances for all values of p. Moreover, the success rate of the dynamic verifier was at least as the success rate of the fixed verifier that achieved the highest success rate. The quality of solution and running time of the dynamic verifier and the fixed verifier were similar for instances that could be solved by both. The dynamic verifier performs better but it is more complicated. Hence there is a tradeoff.

Conclusions and Future work

We studied a new form of robustness: p-robust, and proposed a greedy CBS-based algorithm for finding a p-robust plan with two possible verifiers that have an internal tradeoff. Possible lines of future work, including integrating p-robust plans with execution policies, as suggested by Ma et al. (2017) and better approximating the real P_0 probability.

References

Atzmon, D.; Stern, R.; Felner, A.; Wagner, G.; Barták, R.; and Zhou, N. F. 2018. Robust multi-agent path finding. In *SOCS*, 2–9.

Ma, H.; Kumar, S.; and Koenig, S. 2017. Multi-agent path finding with delay probabilities. In *AAAI*.

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