Finding All Optimal Solutions in Multi-Agent Path Finding

Shahar Bardugo¹, Dor Atzmon²

¹Ben-Gurion University
²Bar-Ilan University
bshahar@post.bgu.ac.il, dor.atzmon@biu.ac.il

Introduction and Problem Definition

Multi-Agent Path Finding (MAPF) (Stern et al. 2019) aims to find conflict-free paths. MAPF is defined by a tuple \((G, A, S, G)\), where \(G = (V, E)\) is an undirected graph; \(A = (a_1, \ldots, a_k)\) is a list of \(k\) agents; and \(S = (s_1, \ldots, s_k)\) and \(G = (g_1, \ldots, g_k)\) are lists of start and goal vertices. A path \(\pi\) for agent \(a_i\) is a list of vertices from \(s_i\) to \(g_i\). Let \(\pi_i(t)\) be the \(t\)-th vertex in \(\pi_i\). Any two consecutive vertices in \(\pi_i\) must be traversable: \(\forall t: (\pi_i(t), \pi_i(t+1)) \in E\). Two paths \(\pi_i\) and \(\pi_j\) conflict if the two agents are simultaneously at the same vertex \((\exists t: \pi_i(t) = \pi_j(t))\) or if the agents simultaneously switch vertices \((\exists t: \pi_i(t) = \pi_j(t+1) \land \pi_i(t+1) = \pi_j(t))\). A plan \(\Pi = (\pi_1, \ldots, \pi_k)\) is a list of paths. The cost \(C(\Pi)\) of plan \(\Pi\) equals the sum of the costs of its paths \((= \sum_{\pi_i \in \Pi} C(\pi_i))\). A solution is a conflict-free plan (any two paths do not conflict). An optimal solution has the lowest cost among all solutions. Consider the problem instance in Fig. 1(a). One of the two agents must wait to avoid a conflict. Therefore, four optimal solutions exist. In this paper, we aim to find all optimal solutions in MAPF. We discuss the representation of all optimal solutions, propose algorithms for finding them, and compare them experimentally.

Representing All Optimal Solutions

We suggest three ways to represent all optimal solutions.

Maintaining All Solutions (MAS). The simplest way for this purpose is to maintain a set of all optimal solutions. To the instance in Fig. 1(a), MAS maintains all four solutions.

Shared state-space MDD (SMDD). An MDD (Sharon et al. 2015) is a data structure that represents multiple paths. In a shared state-space of multiple agents, each state contains a vertex for each agent. An MDD in this state-space (denoted SMDD) represents all optimal MAPF solutions. Fig. 1(b) presents the SMDD to the problem instance in Fig. 1(a); every path represents an optimal MAPF solution.

Multiple MDDs (MMDD). MMDD represents all solutions by a set of MDDs. Every MDD\(_i\) \(\in\) MMDD contains an MDD for each agent. Fig. 1(c) illustrates an MMDD to the problem instance in Fig. 1(a). For any MDD\(_i\) \(\in\) MMDD, any permutation of paths is an optimal solution.

A*-based Approach (A\(_*_\)AS)

Adapting A* (Hart, Nilsson, and Raphael 1968) to find All Solutions (A\(_*_\)AS) is straightforward: (1) instead of returning a single solution, we return a set of solutions; (2) A* halts when the first solution is found, while A\(_*_\)AS halts after the last solution is found; and (3) in the duplicate detection of A\(_*_\)AS, we do not prune nodes with the same cost.

To find all optimal MAPF solutions, we execute A\(_*_\)AS in the state space where any state \(s\) contains vertices for all agents; the start and goal states are the start and goal vertices of the agents; and neighboring state \(s'\) of state \(s\) is every permutation in which the vertices of each agent \(a_i\) are traversable (besides the permutation where all agents wait). Also, states where agents conflict are pruned.

CBS-based Approach (CBS\(_AS\) and CBS-M\(_AS\))

Conflict-Based Search (CBS) (Sharon et al. 2015) is an optimal MAPF algorithm. A constraint \((a_i, x, t)\) prohibits agent \(a_i\) from occupying vertex/edge \(x\) at timestep \(t\). We propose CBS\(_AS\) and CBS-M\(_AS\). Both algorithms’ high-level constructs a similar CT (presented once, in Alg. 1). The main difference is the CT leaves they return. We first describe CBS\(_AS\) (Alg. 1), which calls its high level in line 2.

Each CT node \(N\) contains constraints \(N\).constrains; a plan \(N\).II that satisfies \(N\).constrains; the cost \(N\).cost of plan \(N\).II; and MDDs for all agents \(N\).MDDs that satisfies \(N\).constrains. The high level starts by initializing OPEN, MMDD, UB, and root, and inserting root into OPEN (lines 5-7). The CT node \(N\) with the lowest cost is extracted from OPEN (lines 9). If \(N\).cost exceeds UB, MMDD is returned (lines 10-11). Otherwise, we check if \(N\) is a solution (line
Algorithm 1: CBS

1. CBS (MAPF problem instance instance)
   2. MMDD = HighLevel (instance)
   3. return CreateSMDD (MMDD)

Definition 1: HighLevel (MAPF problem instance instance)

4. Init OPEN, MMDD, UB = ∞
5. Init root with an initial path and no constraints
6. Insert root into OPEN
7. while OPEN is not empty do
   8. Extract N from OPEN // lowest cost
   9. if N.cost > UB then
      10. return MMDD
   11. if IsSolution (N) then
      12. UB = MMDD \cup N.MDDs
      13. MMDD = N.MDDs
      14. continue
      15. ⟨ai, aj, x, t⟩ = GetConflict (N)
      16. N_i = GenerateChild (N, {⟨ai, x, t⟩})
      17. N_j = GenerateChild (N, {⟨aj, x, t⟩})
      18. Insert N_i and N_j into OPEN
      19. return MMDD

Algorithm 2: CBS

1. CBS (MAPF problem instance instance)
2. return HighLevel (instance)
3. GetConflict (CT Node N)
4. return Conflict (⟨ai, aj, x, t⟩) in N.MDDs
5. IsSolution (CT Node N)
6. if N.MDDs is conflict free then
7. return true
8. return false

Experimental Study

We experimented on 8 \times 8 empty grids (empty-8-8) and 32 \times 32 room grids (room-32-32-4), publicly available in the MovingAI repository (Stern et al. 2019). We ran each algorithm on 25 problem instances containing k = \{2, 3, \ldots \} agents. We set the time limit to one minute and measured the success rate and average time (in seconds). The results of this experiment are presented in Fig. 2. As expected, A_{AS} solved problem instances of the smaller empty grid only with a small number of agents and, in the room grids, it did not solve any of the problem instances. In both maps, CBS-M_{AS} outperformed CBS_{AS} and solved problem instances with more agents. CBS-M_{AS} achieves the best results, in terms of runtime, and outperforms both A_{AS} and CBS_{AS}.

References

