Finiding All Optimal Solutions in Multi-Agent Path Finding

Shahar Bardugo[1], **Dor Atzmon**[2]

[1]Ben-Gurion University
[2]Bar-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 $\langle \mathcal{G}, A, S, G\rangle$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an undirected graph; $A = (a_1, \dots, a_k)$ is a list of k agents; and $S = (s_1, \dots, s_k)$ and $G = (g_1, \dots, g_k)$ are lists of start and goal vertices. A *path* π_i 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 π_i. Any two consecutive vertices in π_i must be traversable: $\forall t : (\pi_i(t), \pi_i(t{+}1)) \in \mathcal{E}$. Two paths π_i and π_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 ($\exists t : \pi_i(t) = \pi_j(t+1) \wedge \pi_i(t+1) = \pi_j(t)$). A *plan* $\Pi = (\pi_1, \dots, \pi_k)$ is a list of paths. The cost $C(\Pi)$ of plan Π 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 $MDDs_i \in MMDD$ contains an MDD for each agent. Fig. 1(c) illustrates an $MMDD$ to the problem instance in Fig. 1(a). For any $MDDs_i \in MMDD$, any permutation of paths is an optimal solution.

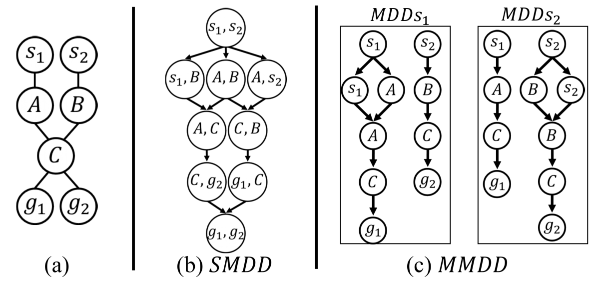

Figure 1: (a) Problem instance. (b,c) Solution representation.

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 $\langle a_i, x, t\rangle$ 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 $N.constraints$; a plan $N.\Pi$ that satisfies $N.constraints$; the cost $N.cost$ of plan $N.\Pi$; and $MDDs$ for all agents $N.MDDs$ that satisfies $N.constraints$. 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

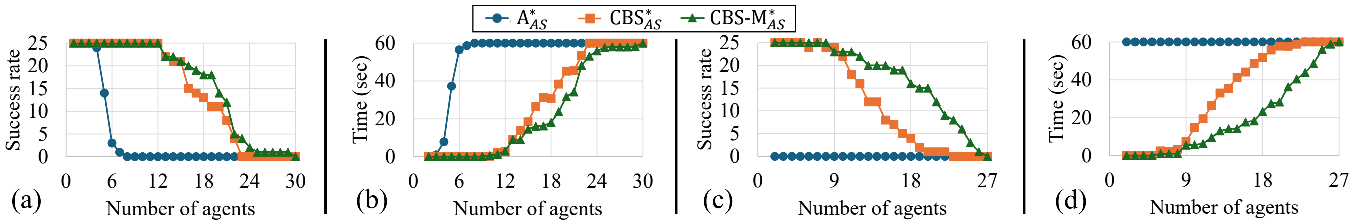

Figure 2: Success rate and time (sec) for A^*_{AS}, CBS_{AS}, and CBS-M_{AS} on 8×8 empty grids (a,b) and 32×32 room grids (c,d).

Algorithm 1: CBS_{AS}

1 **CBS_{AS} (MAPF problem instance** $instance$**)**
2     $MMDD$ = HighLevel ($instance$)
3   **return** CreateSMDD ($MMDD$)
4 **HighLevel(MAPF problem instance** $instance$**)**
5   Init OPEN, $MMDD$, $UB = \infty$
6   Init *root* with an initial plan and no constraints
7   Insert *root* into OPEN
8   **while** OPEN *is not empty* **do**
9   Extract N from OPEN // *lowest cost*
10   **if** $N.cost > UB$ **then**
11   **return** $MMDD$
12   **if** *IsSolution*(N) **then**
13             $MMDD = MMDD \cup N.MDDs$
14             $UB = N.cost$
15   continue
16         $\langle a_i, a_j, x, t\rangle$ = GetConflict(N)
17   N_i = GenerateChild(N, $\{\langle a_i, x, t\rangle\}$)
18   N_j = GenerateChild(N, $\{\langle a_j, x, t\rangle\}$)
19   Insert N_i and N_j into OPEN
20   **return** $MMDD$
21 **GenerateChild (Node** N**, Constraints** C**)**
22     $N'.constraints = N.constraints \cup C$
23     $N'.\Pi = N.\Pi$; $N'.MDDs = N.MDDs$
24   Update $N'.\Pi$ to satisfy $N'.constraints$
25   Update $N'.MDDs$ to satisfy $N'.constraints$
26     $N'.cost = C(N.\Pi)$
27   **return** N'
28 **GetConflict(CT Node** N**)**
29   **return** Conflict $\langle a_i, a_j, x, t\rangle$ in $N.\Pi$
30 **IsSolution(CT Node** N**)**
31   **if** $N.\Pi$ *is conflict free* **then**
32   **return** true
33   **return** false

Algorithm 2: CBS-M_{AS}

1 **CBS-M_{AS} (MAPF problem instance** $instance$**)**
2   **return** HighLevel ($instance$)
3 **GetConflict(CT Node** N**)**
4   **return** Conflict $\langle a_i, a_j, x, t\rangle$ in $N.MDDs$
5 **IsSolution(CT Node** N**)**
6   **if** $N.MDDs$ *is conflict free* **then**
7   **return** true
8   **return** false

12). In CBS_{AS}, N is a solution if $N.\Pi$ is conflict-free (lines 31-32). If it is, N's $MDDs$ ($N.MDDs$) is added to $MMDD$, UB is updated, and we continue to the next CT node (lines 13-15). If N is not a solution, a conflict $\langle a_i, a_j, x, t\rangle$ is chosen (line 16). In CBS_{AS}, this conflict is found in $N.\Pi$ (line 29). The conflict is resolved by generating two CT nodes N_i and N_j, constraining each of the conflicting agents, and inserting the nodes into OPEN (lines 17-19).

When the high level halts, it returns $MMDD$. However, the $MDDs$ may still contain conflicts. Thus, CBS_{AS} merges the set of all $MDDs$ into an $SMDD$ and returns it (line 3).

CBS-M_{AS} (Alg. 2) resolves a new type of conflict, an $MDDs$ conflict (line 4). An $MDDs$ conflict $\langle a_i, a_j, x, t\rangle$ exists if both MDD_i and MDD_j contain vertex/edge x at timestep t. It is resolved similarly to the way a standard conflict is resolved. In CBS-M_{AS}, a CT node N is a solution if it does not contain any $MDDs$ conflict (lines 6-8). CBS-M_{AS} resolves $MDDs$ conflicts so its $MMDD$ only represents valid solutions, and can be returned as is (line 2).

Experimental Study

We experimented on 8×8 empty grids (empty-8-8) and 32×32 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 $k = \{2, 3, \dots\}$ 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

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

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.; Barták, R.; and Boyarski, E. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *SoCS*, 151–159.