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

Multi-Agent Path Finding for Large Agents[*]

Jiaoyang Li
CS Department
Univ. of Southern California
jiaoyanl@usc.edu

Pavel Surynek
Faculty of Information Technology
Czech Technical University
pavel.surynek@fit.cvut.cz

Ariel Felner
SISE Department
Ben-Gurion University
felner@bgu.ac.il

Hang Ma
T. K. Satish Kumar
Sven Koenig
Univ. of Southern California

Abstract

Multi-Agent Path Finding (MAPF) has been widely studied in the AI community. For example, Conflict-Based Search (CBS) is a state-of-the-art MAPF algorithm based on a two-level tree-search. However, previous MAPF algorithms assume that an agent occupies only a single location at any given time, e.g., a single cell in a grid. This limits their applicability in many real-world domains that have geometric agents in lieu of point agents. In this paper, we formalize and study MAPF for large agents that considers the shapes of agents. We present a generalized version of CBS, called Multi-Constraint CBS (MC-CBS), that adds multiple constraints (instead of one constraint) for an agent when it generates a high-level search node. Experimental results show that all MC-CBS variants significantly outperform CBS. The best variant also outperforms EPEA* (a state-of-the-art A*-based MAPF solver) in all cases and MDD-SAT (a state-of-the-art reduction-based MAPF solver) in some cases.

Introduction and Problem Definition

In robotics and computer games, one has to find collision-free paths for multiple agents operating in a common environment. This has led to the study of Multi-Agent Path Finding (MAPF) in the AI community, where we are required to find a path for each agent from its given start vertex to its given goal vertex on a given graph such that no two agents collide with each other at any given time. Although previous MAPF algorithms have found some real-world applications, they are based on one simplistic assumption that limits their applicability. This assumption is to ignore the shape of agents and consider them as point agents, which occupy exactly one point at any time. In reality, agents are geometric in nature with definite shapes. Therefore, they typically occupy a set of points at any time. We refer to such agents as *large agents* and formally study *Multi-Agent Path Finding for Large Agents* (LA-MAPF) which takes into consideration the shapes of agents.

We formalize LA-MAPF as follows: We are given an undirected graph $G = (V, E)$ embedded in a d-dimensional Euclidean space (usually $d = 2, 3$). Each vertex $v \in V$ is a point in the Euclidean space that is specified by its coordinates. We are also given a set of k agents $\{a_1 \dots a_k\}$ with unique start and goal vertices, and each agent has a fixed geometric shape around a reference point that cannot undergo transformations like rotations. We say that an agent is at a vertex v when its reference point is at vertex v, and we say that an agent traverses an edge (u, v) when its reference point moves along edge (u, v). At each discrete timestep t, an agent can either *wait* at its current vertex or *move* to an adjacent vertex. Both wait and move actions have unit costs unless the agent terminally waits at its goal vertex.

In MAPF, a conflict between two agents is either a vertex conflict, where two agents are at the same vertex at the same timestep, or an edge conflict, where two agents traverse the same edge in opposite directions at the same timestep. In LA-MAPF, however, agents could collide when they are in close proximity with each other. Therefore, we generalize the definitions of conflicts. We define a *vertex conflict* as a five-element tuple $\langle a_i, a_j, u, v, t\rangle$, where the shapes of a_i and a_j overlap if a_i is at vertex u and a_j is at vertex v at the same timestep t. Similarly, we define an *edge conflict* as a seven-element tuple $\langle a_i, a_j, u_1, u_2, v_1, v_2, t\rangle$, where the shapes of a_i and a_j overlap if a_i moves from vertex u_1 to u_2 and a_j moves from vertex v_1 to v_2 at the same timestep t. We focus here on resolving vertex conflicts, except for the experimental section, since edge conflicts can be handled analogously. Our task is to find a set of conflict-free paths that move all agents from their start vertices to their goal vertices with minimum sum of costs of all paths.

Conflict-Based Search (CBS)

CBS (Sharon et al. 2015) is a two-level tree-search algorithm that is complete and optimal for MAPF. Its high level performs a best-first search on a binary *constraint tree* (CT). Each CT node contains a set of constraints, where a *constraint* $\langle a_i, u, t\rangle$ prohibits agent a_i from being at vertex u at timestep t, and a set of paths for all agents that satisfy all constraints. The *cost* of the CT node is the sum of costs of all paths. The CT root node contains an empty set of con-

[*]This paper is a short version of (Li et al. 2019). The research at the University of Southern California was supported by the National Science Foundation (NSF) under grant numbers 1409987, 1724392, 1817189 and 1837779 as well as a gift from Amazon. The research was also supported by the United States-Israel Binational Science Foundation (BSF) under grant number 2017692 and the Czech Science Foundation (GACR) under grant number 19-17966S.

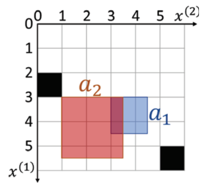

Figure 1: A 4-neighbor 2D grid with square-shaped agents, where lines represent edges, intersection points represent vertices, and black cells represent obstacles. The reference point is the top-left corner of the agent.

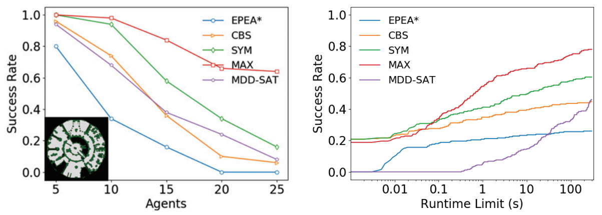

Figure 2: Success rates (= %solved instances) for different numbers of agents within 5 minutes (left) and for all numbers of agents within different runtime limits (right).

straints and a set of individual shortest paths. When the high level expands a CT node N, it first checks for conflicts in its paths. If they are conflict-free, then N is a goal node and CBS returns its paths. Otherwise, the high level resolves a conflict $\langle a_i, a_j, u, v, t\rangle$ ($u = v$ in MAPF) by *splitting* N into two child CT nodes, N_1 and N_2, that inherit all constraints and paths from N. The high level also adds a new constraint $\langle a_i, u, t\rangle$ to N_1 and a new constraint $\langle a_j, v, t\rangle$ to N_2 and then runs a low-level search for both child nodes to find shortest paths for all agents that satisfy the new set of constraints in each of them. With two child CT nodes per conflict, CBS guarantees optimality by exploring both ways of resolving each conflict.

CBS can be directly adapted to LA-MAPF by considering generalized conflicts. However, the resulting version of CBS is very inefficient for large agents. For example, in Figure 1, a_1 tries to move from $(3, 0)$ to $(3, 4)$, and a_2 tries to move from $(0, 1)$ to $(3, 1)$. In the CT root node, both agents follow their individual shortest paths and have conflicts at timesteps 1, 2 and 3. The drawing is a snapshot that shows their conflict at timestep 3. If CBS chooses to resolve this conflict, then the right child node has a new constraint $\langle a_2, (3, 1), 3\rangle$. a_2 is forced to wait for one timestep and thus stays at vertex $(2, 1)$ at timestep 3. However, a_2 then still conflicts with a_1 at timestep 3 in this child node. The conflict between a_1 and a_2 at timestep 3 is therefore not resolved by this split. Although it will be resolved eventually, many intermediate CT nodes are generated.

Multi-Constraint CBS (MC-CBS)

The above observations motivate the idea of adding multiple constraints in a single CT node expansion. We present a new algorithm, MC-CBS, which adds multiple constraints for the same timestep to child nodes in order to resolve multiple related conflicts in a single CT node expansion, which resembles lookahead reasoning at the high level and can result in smaller CTs, thus making the search more efficient.

We say that two constraints for a_i and a_j respectively are *mutually disjunctive* iff any pair of conflict-free paths for a_i and a_j satisfies at least one of the two constraints, i.e., there do not exist two conflict-free paths such that both constraints are violated. In particular, the constraints that CBS adds to two child nodes are always mutually disjunctive. We say that two sets of constraints are *mutually disjunctive* iff each constraint in one set is mutually disjunctive with each constraint in the other set.

When MC-CBS resolves a conflict $\langle a_i, a_j, u, v, t\rangle$ in a CT node N, it generates two child nodes with N's constraint set and additional constraint sets, C_1 and C_2, respectively: (1) C_1 and C_2 include the *core constraints* that CBS uses to resolve the conflict, i.e., $\langle a_i, u, t\rangle \in C_1$ and $\langle a_j, v, t\rangle \in C_2$, and (2) C_1 and C_2 are enhanced with other constraints that ensure that C_1 and C_2 remain mutually disjunctive.

MC-CBS is complete and optimal (Li et al. 2019). We present three approaches to choosing such constraint sets:

1. ASYM adds a constraint set of size one to the left child node that prohibits agent a_i from being at its current vertex v at timestep t and a large constraint set to the right child node that prohibits agent a_j from being at any vertex at timestep t where it could collide with a_i.
2. SYM chooses a point p in the Euclidean space that is inside the overlap area, and then adds one constraint set to each child node. The constraint set blocks all vertices that the agent could be at while including p in its shape.
3. MAX tries all pairs of constraint sets that satisfy constraints (1) and (2) and then chooses the pair where the child nodes have the highest possible costs.

Experimental Results

We compare the three MC-CBS variants (i.e., ASYM, SYM and MAX) with CBS, EPEA*, and MDD-SAT on instances with square-shaped agents of different sizes randomly chosen from $\{2.5, 3.5, 4.5\}$. We used a 4-neighbor grid lak503d from (Sturtevant 2012). We used 50 instances with randomly generated start vertices and goal vertices for each number of agents. Figure 2 presents the results. ASYM always performs slightly worse than SYM, and thus we didn't plot it in the figures. But both of them always perform better than CBS. MAX significantly outperforms all other algorithms in all cases except when the runtime limit is less than 0.1 s.

References

Li, J.; Surynek, P.; Felner, A.; Ma., H.; Kumar, T. K. S.; and Koenig, S. 2019. Multi-agent path finding for large agents. In *AAAI Conference on Artificial Intelligence*.

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

Sturtevant, N. R. 2012. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games* 4(2):144–148.