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
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 $k$ of agents $\{a_1, \ldots, a_k\}$ 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 $(a_i, a_j, u, v, t)$, 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 $(a_i, a_j, u_1, u_2, v_1, v_2, t)$, 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 $(a_i, u, t)$ 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-
then the right child node has a new constraint. If CBS chooses to resolve this new constraint, they will run 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, a1 tries to move from (3, 0) to (3, 4), and a2 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.

When MC-CBS resolves a conflict \( \langle x, y, u, v, t \rangle \) 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.

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, 3, 5, 4, 5\}. We used a 4-neighbor grid lsk503d 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