The Thirteenth International Symposium
on Combinatorial Search (SoCS 2020)

New Techniques for Pairwise Symmetry Breaking in Multi-Agent Path Finding[*]

Jiaoyang Li,[1] Graeme Gange,[2] Daniel Harabor,[2] Peter J. Stuckey,[2] Hang Ma,[3] Sven Koenig[1]
[1]University of Southern California, [2]Monash University, [3]Simon Fraser University
jiaoyanl@usc.edu, {graeme.gange, daniel.harabor, peter.stuckey}@monash.edu, hangma@sfu.ca, skoenig@usc.edu

Abstract

We consider two new classes of pairwise path symmetries which appear in the context of Multi-Agent Path Finding (MAPF). The first of them, *corridor symmetry*, arises when two agents attempt to pass through the same narrow passage in opposite directions. The second, *target symmetry*, arises when the shortest path of one agent passes through the target location of a second agent after the second agent has already arrived at it. We propose to break these symmetries using specialized constraints while preserving optimality. We experimentally show that our techniques can significantly speed up Conflict-Based Search, a state-of-the-art MAPF algorithm.

Introduction

Multi-Agent Path Finding (MAPF) is specified by an undirected graph and a set of m agents $\{a_i | i = 1, \dots, m\}$, each with a start vertex s_i and a target vertex g_i. At each discrete timestep, an agent can either move to an adjacent vertex or wait at its current vertex. A *path* for agent a_i is a sequence of vertices which are adjacent or identical (indicating a wait action), starting at s_i and ending at g_i. Agents remain at their target vertices after they complete their paths. A *conflict* happens when two agents are at the same vertex at the same timestep or traverse the same edge in opposite directions at the same timestep. Our task is to find a set of paths with the minimum sum of path lengths that move all agents from their start vertices to their target vertices without conflicts.

Conflict-Based Search (CBS) (Sharon et al. 2015) is a well-known algorithm for solving MAPF optimally. It performs a best-first search on a binary *constraint tree* (CT). Each CT node contains a set of constraints that are used to coordinate agents, a set of shortest paths, one for each agent, that satisfy these constraints and a cost that equals the sum of the path lengths. The root CT node contains an empty set of constraints. When expanding a CT node N, CBS checks for conflicts in the paths of N. If there are none, N is a goal CT node, and CBS terminates. Otherwise, CBS branches by choosing a conflict and resolving it by splitting N and generating two child CT nodes. In each child CT node, one agent from the conflict is prohibited from using the conflicting vertex or edge at the conflicting timestep by way of an additional constraint. The current path of this agent does not satisfy the new constraint and is thus replanned. All other paths remain unchanged.

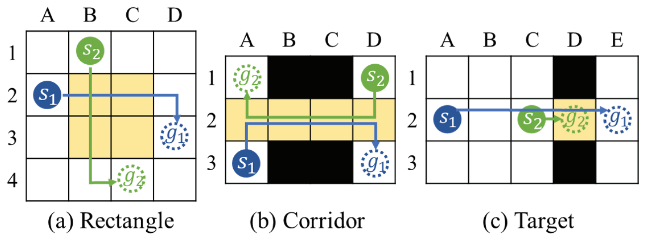

Figure 1: Examples of rectangle, corridor and target symmetries between two agents on 4-neighbor grids.

Recent work (Li et al. 2019) shows that CBS suffers from unacceptable runtimes when two agents are involved in a *rectangle symmetry* on 4-neighbor grids. Figure 1(a) shows an example. Each agent has multiple shortest paths, but any shortest path for one agent conflicts with any shortest path for the other agent in the yellow area. The only resolution is for one of the agents to wait or take a detour. However, to generate a longer path for one agent, CBS has to branch multiple times and try many combinations of shortest paths.

In this work, we explore two new classes of pairwise symmetries, namely *corridor symmetry* and *target symmetry*. Like their rectangle counterpart, each one describes a specific situation that arises in MAPF, and each symmetry-breaking technique preserves the optimality of CBS. Unlike their rectangle counterpart, both of them are applicable to MAPF on arbitrary graphs rather than just 4-neighbor grids.

Corridor Symmetry

A corridor symmetry occurs when two agents traverse a corridor in opposite directions at the same time, where a *corridor* is a chain of connected vertices C_0, each of degree 2, together with two endpoints e_1 and e_2 connected to C_0. Figure 1(b) shows an example, where the corridor is highlighted in yellow. CBS detects a conflict on edge (B2, C2) at timestep 3. Each agent has many shortest paths that satisfy

[*]This paper is a short version of (Li et al. 2020). 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 constraint to avoid edge (B2, C2) at timestep 3 (i.e., involve one wait action before timestep 3), but each of them remains in conflict with the path of the other agent. CBS has to branch at least four times to find conflict-free paths in such a situation and has to branch even more times to prove their optimality. Moreover, as the *corridor length* k (i.e, the distance between its two endpoints) increases, the number of expanded CT nodes grows exponentially as 2^{k+1}. We therefore propose a symmetry-breaking technique that can resolve corridor symmetries efficiently.

Consider a corridor of length k with endpoints e_1 and e_2. Assume that agent a_1 traverses the corridor from e_2 to e_1 and agent a_2 traverses the corridor from e_1 to e_2. They conflict with each other inside the corridor. Let t_i be the earliest timestep when agent a_i can reach e_i ($i = 1, 2$). If we prioritize agent a_1 and let agent a_2 wait, then the earliest timestep when agent a_2 can start to traverse the corridor from e_1 is t_1+1. Therefore, the earliest timestep when agent a_2 can reach e_2 is $t_1 + 1 + k$. But if there exist *bypasses* such that agent a_2 can reach e_2 without traversing the corridor, then the earliest timestep when agent a_2 can reach e_2 is $\min(t'_2, t_1 + 1 + k)$, where t'_2 is the earliest timestep when agent a_2 can reach e_2 without traversing the corridor. Similarly, if we prioritize agent a_2, then the earliest timestep when agent a_1 can reach e_1 is $\min(t'_1, t_2 + 1 + k)$, where t'_1 is the earliest timestep when agent a_1 can reach e_1 without traversing the corridor. In other words, any paths of agent a_1 that reach e_1 before or at timestep $\min(t'_1 - 1, t_2 + k)$ must conflict with any paths of agent a_2 that reach e_2 before or at timestep $\min(t'_2 - 1, t_1 + k)$. Thus, to resolve the corridor symmetry, we generate two child CT nodes with two range constraints $\langle a_1, e_1, [0, \min(t'_1 - 1, t_2 + k)]\rangle$ and $\langle a_2, e_2, [0, \min(t'_2 - 1, t_1 + k)]\rangle$, one for each child CT node, where a *range constraint* $\langle a_i, v, [t_{min}, t_{max}]\rangle$ prohibits agent a_i from being at vertex v at any timestep from timestep t_{min} to timestep t_{max}.

Target Symmetry

A target symmetry occurs when one agent traverses the target vertex of a second agent after the second agent has already arrived at it and remains there. Figure 1(c) shows an example. Agent a_2 arrives at its target vertex D2 at timestep 1, but an unavoidable vertex conflict occurs with agent a_1 at vertex D2 at timestep 3. To resolve this conflict, CBS generates two child CT nodes. In the left child CT node, CBS prohibits agent a_2 from being at vertex D2 at timestep 3 and finds a new path [C2, C3, C3, C2, D2] for it, which does not conflict with agent a_1. The cost of this CT node is three larger than the cost of the root CT node. In the right child CT node, CBS prohibits agent a_1 from being at vertex D2 at timestep 3 and finds a new path that arrives at vertex D2 at timestep 4. The cost of this CT node is one larger than the cost of the root CT node. However, this new path produces a further conflict with agent a_2 at vertex D2 at timestep 4. Although the left child CT node contains conflict-free paths, CBS has to split the right child CT nodes repeatedly to constrain agent a_1 before eventually proving that the paths of the left child CT node is optimal. Moreover, as the distance k between vertices s_1 and g_2 in Figure 1(c) increases, the number of expanded CT nodes grows linearly in k. Although, this may not seem too problematic, only one of the leaf CT nodes actually resolves the conflict. Later, when other conflicts occur elsewhere on the map, each of the leaf CT nodes will be further fruitlessly expanded. For example, with m copies of the problem (resulting in a $2m$-agent instance), the number of expanded CT nodes increases exponentially in m.

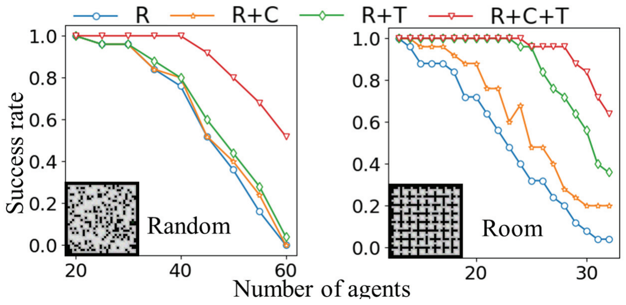

Figure 2: Success rates. R, C and T are short for rectangle, corridor and target reasoning, respectively.

The key to resolving target symmetry is to reason about the path length of an agent directly. Suppose agent a_2 arrives at its target vertex g_2 at timestep t' and remains there. Agent a_1 traverses vertex g_2 at timestep t ($t \geq t'$). We resolve this conflict by branching on the path length l_2 of agent a_2 using two *length constraints* $l_2 > t$ and $l_2 \leq t$, one for each child CT node. In the left child CT node, $l_2 > t$ forces agent a_2 to complete its path after timestep t. Thus, we need to replan the path for agent a_2. Its path length increases from its current value t' to at least $t + 1$. In the right child CT node, $l_2 \leq t$ forces agent a_2 to reach its target vertex g_2 and remain there before or at timestep t, which also prohibits any other agent from being at vertex g_2 at or after timestep t. We do not need to replan the path for agent a_2 since its current path is no longer than t. Nevertheless, we need to replan the paths for agent a_1 and all other agents that traverse vertex g_2 at or after timestep t.

Experiments

We implement CBS with rectangle reasoning (i.e., CBSH-RM in (Li et al. 2019)), corridor reasoning and target reasoning in C++. The experiments are conducted on a 2.80 GHz Intel Core i7-7700 laptop with 8 GB RAM. Figure 2 plots the *success rates*, i.e., the percentages of solved instances within one minute. See (Li et al. 2020) for more results.

References

Li, J.; Harabor, D.; Stuckey, P. J.; Ma, H.; and Koenig, S. 2019. Symmetry-breaking constraints for grid-based multi-agent path finding. In *AAAI*, 6087–6095.

Li, J.; Gange, G.; Harabor, D.; Stuckey, P. J.; Ma, H.; and Koenig, S. 2020. New techniques for pairwise symmetry breaking in multi-agent path finding. In *ICAPS*.

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