Nested ECBS for Bounded-Suboptimal Multi-Agent Path Finding

Shao-Hung Chan[1] , **Jiaoyang Li**[1] , **Daniel Harabor**[2] , **Peter J. Stuckey**[2] , **Graeme Gange**[2] , **Liron Cohen**[1] , **Sven Koenig**[1]

[1]University of Southern California, USA
[2]Monash University, Australia
{shaohung,jiaoyanl}@usc.edu, {daniel.harabor,peter.stuckey,graeme.gange}@monash.edu, {lironcoh,skoenig}@usc.edu

Abstract

Multi-Agent Path Finding (MAPF) is the problem of finding collision-free paths for multiple agents on a map. Conflict-Based Search (CBS) is a powerful, complete, and optimal MAPF solver, while Enhanced CBS (ECBS) improves the efficiency of CBS by only guaranteeing a bounded-suboptimal solution. Both MAPF solvers suffer from the weakness of repeatedly resolving the same collisions between the same agents. Merging agents into meta-agents and planing their paths in the joint state space can be used to overcome this problem. However, a joint-state-space MAPF solver makes resolving collisions within meta-agents inefficient. In this paper, we therefore propose Nested ECBS (NECBS), a nested architecture based on ECBS, where collisions within meta-agents are resolved with ECBS. NECBS preserves the important properties of ECBS, namely its completeness and bounded-suboptimality. Empirically, NECBS has a higher success rate than ECBS and its state-of-the-art variants for a runtime limit of 5 minutes.

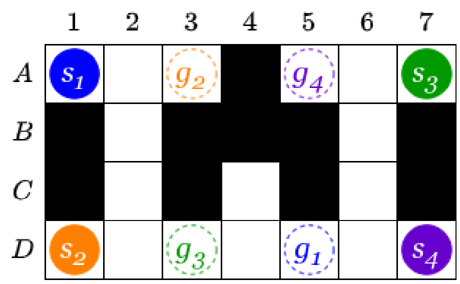


Figure 1: An illustrative 4-agent MAPF instance. Each start vertex s_i of agent a_i is shown as a solid circle, while its goal vertex g_i is shown as a hollow circle in the same color.

1 Introduction

Multi-Agent Path Finding (MAPF) is the problem of finding collision-free paths for multiple agents moving on a map. A common objective of MAPF is to minimize the sum of the travel times of the agents. MAPF has many real-world applications, such as autonomous aircraft-towing vehicles [Morris *et al.*, 2016], office robots [Veloso *et al.*, 2015], video game characters [Ma *et al.*, 2017], and quadrotor swarms [Hönig *et al.*, 2018]. We will discuss the problem definition of MAPF in Section 2. Conflict-Based Search (CBS) [Sharon *et al.*, 2015] is a successful MAPF solver for solving MAPF optimally in low-contention environments. CBS optimistically plans paths for agents independently and resolves collisions using best-first search in the space of collision resolutions. However, CBS and its variants run into difficulties when multiple sets of agents collide frequently with one another, which leads to resolving the same collisions between the same agents repeatedly. We call this issue the *repeated replanning problem*. We will discuss some existing MAPF solvers and their limitations in Section 3

Consider the MAPF instance in Figure 1. The agents on the left (a_1 and a_2) must coordinate the use of the narrow vertical corridor $\{A2, B2, C2, D2\}$. One of them must take a delay of at least 4 timesteps at its start vertex in order to avoid colliding with the other one. Similarly, one of the agents on the right (a_3 or a_4) must take a delay of at least 4 timesteps. However, without dedicated corridor reasoning techniques [Li *et al.*, 2020], CBS cannot discover this issue immediately and instead must investigate many paths with delays of fewer than 4 timesteps, eventually concluding that delays of 4 timesteps are inevitable to avoid collisions.

One successful MAPF solver for preventing agents from frequently colliding with one another is Enhanced CBS (ECBS) [Barer *et al.*, 2014], which uses *focal search* [Pearl and Kim, 1982] to find bounded-suboptimal collision-free paths. As long as the bound is sufficiently loose, ECBS can quickly find collision-free paths where agents are allowed to take some delays, as it does not have to prove that solutions of lower costs do not exist. In Figure 1, for example, ECBS can speed up the search by delaying either agent a_1 or agent a_2 and by delaying either agent a_3 or agent a_4 within a user-specified bound. However, if the bound is not sufficiently loose, the repeated replanning problem remains a weakness of ECBS.

Another MAPF solver for preventing agents from frequently colliding with one another is Meta-Agent CBS (MA-CBS) [Sharon *et al.*, 2015], which merges a set of agents after observing that the agents in the set collide repeatedly with one another and then treats them as one *meta-agent*. In the exam-

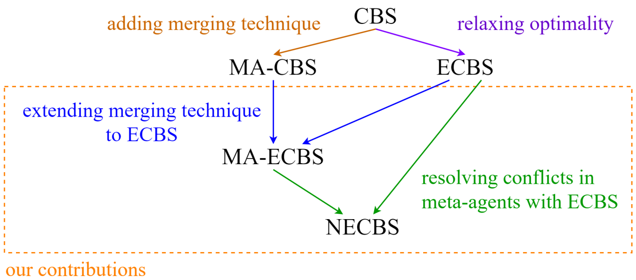


Figure 2: Relationships between CBS-related MAPF solvers. Arrows in the same color correspond to the same improvements. Our contributions include extending the merging technique from CBS to ECBS, resulting in Meta-Agent ECBS (MA-ECBS) and using ECBS to resolve the collisions between agents within the same meta-agent, resulting in Nested ECBS (NECBS).

ple of Figure 1, after we have resolved the collision between agents a_1 and a_3 in the horizontal corridor $\{D3, D4, D5\}$, merged agents a_1 and a_2 into one meta-agent, and merged agents a_3 and a_4 into another meta-agent, we have essentially solved the repeated replanning problem since we can resolve the collisions between the agents in each meta-agent independently. Merging agents a_1 and a_2 allows for their coordination with respect to vertical corridor $\{A2, B2, C2, D2\}$, while merging agents a_3 and a_4 does the same for vertical corridor $\{A6, B6, C6, D6\}$. Since the collision-free paths of all agents in each meta-agent are planned optimally, MA-CBS is guaranteed to find optimal collision-free paths. We extend the merging technique to ECBS and present the resulting bounded-suboptimal MAPF solver MA-ECBS in Section 4. However, both MA-CBS and MA-ECBS resolve the collisions between all agents in the same meta-agent via a *joint-state-space MAPF solver*, which finds collision-free paths by viewing the state space as the cross-product of the possible vertices of each agent. The size of the resulting search space grows exponentially in the number of agents, which makes planning intractable for meta-agents containing many agents.

We thus propose Nested ECBS (NECBS), which is a bounded-suboptimal MAPF solver that combines ECBS and MA-ECBS to address the repeated replanning problem. It uses MA-ECBS but resolves the collisions between the agents in the same meta-agent with ECBS instead of the joint-state-space MAPF solver, thus combining two MAPF solvers to improve the efficiency of ECBS, while still being bounded-suboptimal (discussed in Section 5). The relationships among the various CBS-related MAPF solvers are shown in Figure 2. We also use techniques such as restarting the search right after merging agents, known as *Merge and Restart* (MR) [Boyarski *et al.*, 2015], to improve the efficiency even further. Our empirical results in Section 7 show that NECBS has a higher success rate than ECBS. Furthermore, they show that NECBS with MR (NECBS (MR)) has a 2.75% higher success rate than ECBS (RR), a state-of-the-art version of ECBS, for a runtime limit of 5 minutes.

2 Problem Definition

While there exist a variety of definitions for Multi-Agent Path Finding (MAPF), we use the one in [Stern *et al.*, 2019]. The MAPF problem consists of an undirected graph $G = (V, E)$ and a set of k agents $\{a_1, \ldots, a_k\}$. Each agent a_i has a unique start vertex $s_i \in V$ and a unique goal vertex $g_i \in V$. Time is discretized into timesteps, starting at 0. At every timestep, an agent is allowed to either *move* to an adjacent vertex or *wait* at its current vertex. Also, an agent continues to exist after it has reached its goal vertex.

A path p_i starting at start vertex s_i and ending at goal vertex g_i is a sequence of vertices indicating where agent a_i is at each timestep. In addition, any two adjacent vertices on path p_i are either adjacent in G, meaning that agent a_i *moves*, or identical, meaning that agent a_i *waits*. The *cost* of path p_i is its length, that is, the number of timesteps needed by agent a_i to move from vertex s_i to vertex g_i, ignoring the timesteps when agent a_i terminally waits at vertex g_i. A *collision* (or, equivalently, *conflict*) between two agents belongs to one of two categories. It can be a *vertex conflict* $\langle a_i, a_j, v, t\rangle$, where agents a_i and a_j occupy the same vertex $v \in V$ at the same timestep t. It can also be an *edge conflict* $\langle a_i, a_j, u, v, t\rangle$, where agents a_i and a_j traverse the same edge $(u, v) \in E$ in opposite directions from timestep t to timestep $t+1$. A *solution* is a set of conflict-free paths, one for each agent. An *optimal solution* is a solution with minimum *sum of the costs (SoC)* of the paths. In this paper, all graphs are 4-neighbor grids with vertices corresponding to the unblocked cells and edges corresponding to the connections between adjacent unblocked cells in the four main compass directions.

3 Existing MAPF Solvers

In this section, we present Conflict-Based Search (CBS), Enhanced CBS (ECBS), and Meta-Agent CBS (MA-CBS). All of them rely on a two-level architecture where the low level plans paths for single agents and the high level resolves conflicts between agents.

3.1 Conflict-Based Search (CBS)

Before introducing CBS [Sharon *et al.*, 2015], we define *constraints*, that are used to resolve conflicts between two agents. A vertex conflict $\langle a_i, a_j, v, t\rangle$ can be resolved by prohibiting either agent a_i or agent a_j from occupying vertex v at timestep t, resulting in *vertex constraints* $\langle a_i, v, t\rangle$ and $\langle a_j, v, t\rangle$. Similarly, an edge conflict $\langle a_i, a_j, u, v, t\rangle$ can be resolved by prohibiting agent a_i from traversing the edge from vertex u to vertex v from timestep t to timestep $t+1$ or prohibiting agent a_j from traversing the edge from vertex v to vertex u from timestep t to timestep $t+1$, resulting in *edge constraints* $\langle a_i, u, v, t\rangle$ and $\langle a_j, v, u, t\rangle$.

On the low level, CBS views vertex $v \in V$ and timestep t as a spatio-temporal node $n = (v, t)$ and runs A* to find an optimal path for each agent a_i that satisfies the vertex and edge constraints provided by the high level. To this end, the low-level search uses an open list OPEN and sorts the spatio-temporal nodes in OPEN in increasing order of their f values $f^i(n) = g^i(n) + h^i(n)$, where $g^i(n)$ is the number of timesteps for agent a_i to move from its start vertex s_i to vertex v and

$h^i(n)$ is an admissible heuristic that estimates the cost from vertex v to its goal vertex g_i. The low-level search breaks ties in favor of paths that have the fewest number of conflicts with the paths of other agents.

On the high level, CBS generates a binary *constraint tree* (CT). Each CT node N has two components: (1) a set $N.paths$ of paths for all agents generated by the low-level search, with the path of agent a_i being $N.paths[i]$ and its cost being $|N.paths[i]|$, and (2) a set $N.constraints$ of vertex and edge constraints that coordinate agents to avoid conflicts. The *cost* $N.cost$ of CT node N is the SoC of $N.paths$, that is,

$$N.cost = \sum_{i=1}^{k} |N.paths[i]|. \tag{1}$$

CBS runs a best-first search on the high level by selecting the CT node with the optimal $N.cost$. It begins the search at the *root* CT node, that contains an optimal path for each agent and an empty set of constraints. While expanding CT node N, if there are no conflicts among the paths of CT node N, CBS returns the solution consisting of the paths of CT node N and terminates. Otherwise, it picks one of the conflicts and resolves it by *branching*, also known as splitting CT node N into two child CT nodes. In each child CT node, CBS adds a vertex or edge constraint for one of the conflicting agents to prohibit it from utilizing the conflicted vertex or edge, respectively, at the conflicted timestep. It then performs a low-level search to replan the path of the agent with the added constraint and leaves all other paths unchanged. CBS solves MAPF optimally by performing best-first searches on both levels.

3.2 Enhanced CBS (ECBS)

Enhanced CBS (ECBS) [Barer *et al.*, 2014] uses focal search, rather than best-first search, on both the high and low levels to speed up CBS significantly and provide bounded-suboptimal solutions. On the low level, given a CT node N, ECBS uses an open list OPEN and sorts its spatio-temporal nodes n in increasing order of their f values $f_1^i(n)$, which is identical to function $f^i(n)$ of CBS, to find a path for agent a_i. Let $best^i$ be the node n with the minimum $f_1^i(n)$ value in OPEN and w be the user-specified suboptimality factor. The low-level focal search also uses a focal list FOCAL that contains all spatio-temporal nodes $n = (v,t)$ in OPEN with $f_1^i(n) \le w \cdot f_1^i(best^i)$. The nodes in FOCAL are sorted in increasing order of their different f values $f_2^i(n)$, where $f_2^i(n)$ specifies the number of conflicts of the path of agent a_i with the paths of other agents in $N.paths$ while agent a_i moves from its start vertex s_i to vertex v. The low-level focal search expands a node n with the minimum $f_2^i(n)$ value in FOCAL. Since $f_1^i(n)$ uses an admissible heuristic, $f_1^i(best^i)$ is a lower bound on the cost of the optimal path of agent a_i. Thus, the low-level focal search always returns a path for agent a_i with a cost of at most w times the optimal path cost c_i^*, meaning that,

$$f_1^i(best^i) \le |N.paths[i]| \le w \cdot f_1^i(best^i) \le w \cdot c_i^*. \tag{2}$$

The low-level focal search also returns lower bound $N.lb[i]$ on the cost of the optimal path for agent a_i, which is the $f_1^i(best^i)$ value when the low-level search terminates.

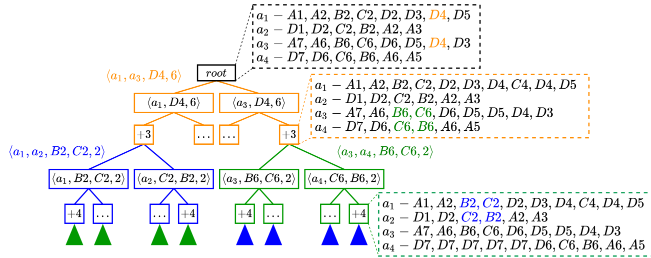


Figure 3: A CT for solving the MAPF instance in Figure 1 with CBS.

On the high level, ECBS runs a focal search on the CT. Given a CT node N, we define its lower bound as $N.LB = \sum_{i=1}^{k} f_1^i(best^i)$. We define the number of conflicts between all pairs of paths in CT node N as $f_2(N)$. Let $LB = \min(N.LB \mid N \in \text{OPEN})$. Since $N.LB$ is a lower bound on the minimum SoC of the solutions below CT node N, LB is a lower bound on the minimum cost, denoted as C^*. The high-level focal search sorts the CT nodes N in OPEN in increasing order of $N.LB$ and adds the CT nodes N with costs of at most $w \cdot LB$ into FOCAL, where w is the same suboptimality factor as used on the low level. ECBS expands the CT node with the minimum f_2 value in FOCAL, namely the CT node with the fewest number of conflicts. Given a MAPF instance with minimum cost C^*, since ECBS only selects CT nodes whose costs are at most $w \cdot LB$ and $LB \le C^*$, it finds a solution whose cost is at most $w \cdot C^*$, that is,

$$LB \le N.cost \le w \cdot LB \le w \cdot C^*. \tag{3}$$

A larger suboptimality factor w or an increase of the lower bound LB during the search typically result in more CT nodes being added to FOCAL, which increases the chance of finding a CT node that contains conflict-free paths, that is, solutions. Thus, given a suboptimality factor, the lower-bound improvement provides a clue about the efficiency of bounded-suboptimal MAPF solvers. We define the lower-bound improvement as

$$LB\ Improvement = \hat{LB} - N_R.LB, \tag{4}$$

where N_R is the root CT node, $N_R.LB$ is the lower bound LB in the beginning of the search, and $\hat{LB}$ is the LB value once a solution has been found.

3.3 Limitations of CBS and ECBS

One of the limitations of CBS and ECBS is the repeated replanning problem. When (E)CBS branches on one conflict and that does not resolve a different conflict as well, it will potentially have to resolve that different conflict in both subtrees. Hence, (E)CBS can resolve the same conflicts many times and consequently repeat a lot of work in different branches of the CT. Consider the MAPF instance shown in Figure 1. One of the possible CTs generated by CBS is shown in Figure 3. The search can be divided into 3 parts: resolving conflicts between agents a_1 and a_3 in the horizontal corridor $\{D3, D4, D5\}$, resolving conflicts between agents a_1 and a_2 in the vertical corridor $\{A2, B2, C2, D2\}$, and resolving conflicts between agents a_3 and a_4 in the vertical corridor

$\{A6, B6, C6, D6\}$. Each conflict resolution results in a subtree drawn in orange, blue, and green, respectively. Some subtrees are simplified as triangles. Each subtree has the following components:

- Each solid-border rectangle with a vertex or edge constraint is an intermediate CT node with the constraint added to the CT node in order to resolve the conflict shown next to it. The part of the tree between the intermediate CT nodes and the leaf CT nodes are not shown.
- Each solid-border square with "$+x$" is a leaf CT node of the subtree in the same color. (It is also the root CT node of the subtree below it.) It contains paths in which the conflicts corresponding to its subtree have been resolved optimally. The value x indicates that the cost of the leaf CT node is x larger than the cost of the root CT node of its subtree.
- Each solid-border square with "..." contains (multiple) leaf CT nodes. It contains paths in which the conflicts corresponding to its subtree have been resolved suboptimally.
- Each dashed-border rectangle contains the paths of the root CT node of the subtree in the same color. The conflicts are highlighted in the same color.

The high-level search in Figure 3 proceeds as follows: CBS first chooses conflict $\langle a_1, a_3, D4, 6\rangle$ at the root CT node and branches accordingly. The CT nodes in which all conflicts between agents a_1 and a_3 in the horizontal corridor $\{D3, D4, D5\}$ have been resolved are shown as the left-most and right-most orange squares. Both of their costs increase by 3 with respect to the root CT node. Then, due to the fact that CBS always selects the CT node in OPEN with the minimum cost, CBS expands the blue and green subtrees rooted at the orange "+3" CT nodes simultaneously. Finally, since the paths of agents a_3 and a_4 of every blue "+4" CT node have conflicts, CBS has to resolve all conflicts between agents a_3 and a_4 in every blue "+4" CT node, resulting in many green subtrees. CBS also has to perform repeated work for each green "+4" CT node, resulting in many blue subtrees. ECBS can reduce the sizes of the subtrees in two ways: (1) The low-level focal search can resolve some of the conflicts by finding bounded-suboptimal paths for the agents, leaving the high level fewer conflicts to resolve; and (2) the high-level focal search can avoid expanding CT nodes as long as their costs are within the suboptimality bound. However, the effectiveness of both ways depends on whether the suboptimality factor w is sufficiently large. In general, the smaller w is, the more likely the repeated replanning problem occurs.

Independence detection (ID) [Standley, 2010] can avoid the repeated replanning problem in some cases. Two sets of agents are *independent* if and only if there exists an optimal solution for each set of agents such that the paths in the two solutions do not conflict [Sharon *et al.*, 2015]. ID attempts to identify independent sets of agents and decompose a MAPF instance into several sub-instances, one for each set of agents. In Figure 1, agents a_1 and a_3 collide with one another in the horizontal corridor $\{D3, D4, D5\}$. If their goal vertices g_1 and g_3 were swapped, then ID could decompose the MAPF instance into two sub-instances, namely the one on the left with agents a_1 and a_2 and the one on the right with agents a_3 and a_4, which would speed up the search since both MAPF sub-instances can be solved separately instead of jointly. However, as the number of agents increases in a MAPF instance, the likelihood of finding independent sets of agents drops rapidly and ID becomes less helpful.

3.4 Meta-Agent CBS (MA-CBS) and the Merging Technique

CBS is inefficient for agents that conflict repeatedly when the high-level search tries to resolve their conflicts. The motivation behind Meta-Agent CBS (MA-CBS) [Sharon *et al.*, 2012; Sharon *et al.*, 2015] is to avoid the resulting repeated replanning problem. MA-CBS groups the agents that conflict repeatedly with one another into a meta-agent and plans conflict-free paths for them in their joint state space, which is known as the *merging* technique. We define a meta-agent as a set of agents, and the *size* of the meta-agent as the number of agents in it. Conflicts between the agents in a meta-agent are called the *internal* conflicts of the meta-agent and are resolved by a joint-state-space MAPF solver when the meta-agent is created. In contrast, conflicts between the agents in different meta-agents are called *external* conflicts between the meta-agents and are resolved by the high-level search. Specifically, we say that meta-agents A_m and A_n have a vertex conflict $\langle A_m, A_n, v, t\rangle$ or an edge conflict $\langle A_m, A_n, u, v, t\rangle$ if and only if there exist two agents $a_i \in A_m$ and $a_j \in A_n$ that have a vertex conflict $\langle a_i, a_j, v, t\rangle$ or an edge conflict $\langle a_i, a_j, u, v, t\rangle$, respectively. MA-CBS uses a $k \times k$ matrix M to record the number of conflicts between any two agents that have been resolved so far. The meta-agents always form a partition of the set of k agents, that is, $A_m \cap A_n = \phi$.

In the beginning, MA-CBS contains k meta-agents, one for each agent, and initializes $M[a_i][a_j]$ with 0 for all agents a_i and a_j. After having selected a CT node for branching, MA-CBS chooses an (external) conflict, either a vertex conflict $\langle A_m, A_n, v, t\rangle$ or an edge conflict $\langle A_m, A_n, u, v, t\rangle$, and increases both $M[a_i][a_j]$ and $M[a_j][a_i]$ by 1 for all agents $a_i \in A_m$ and agents $a_j \in A_n$. MA-CBS uses a user-specified threshold b to determine if two meta-agents are frequently conflicting. If $\sum_{a_i \in A_m} \sum_{a_j \in A_n} M[a_i][a_j] > b$, then MA-CBS considers meta-agents A_m and A_n to be repeatedly conflicting and merges them instead of branching, resulting in a new meta-agent $A_m \cup A_n$. MA-CBS then resolves all internal conflicts of meta-agent $A_m \cup A_n$ with a complete and optimal joint-state-space MAPF solver, such as EPEA* [Goldenberg *et al.*, 2014] or M* [Wagner and Choset, 2011], and reinserts the CT node into OPEN.

If MA-CBS does not merge meta-agents A_m and A_n, then it resolves the chosen conflict by branching. If the chosen conflict is a vertex conflict, then MA-CBS adds meta-agent vertex constraints $\langle A_m, v, t\rangle$ and $\langle A_n, v, t\rangle$ to the two resulting child CT nodes, respectively, where $\langle A_m, v, t\rangle$ represents the set of vertex constraints $\langle a_i, v, t\rangle$ for all agents $a_i \in A_m$ and $\langle A_n, v, t\rangle$ represents the set of vertex constraints $\langle a_j, v, t\rangle$ for all agents $a_j \in A_n$. Similarly, if the chosen conflict is an edge conflict $\langle A_m, A_n, u, v, t\rangle$, then MA-CBS adds meta-agent edge constraints $\langle A_m, u, v, t\rangle$ and $\langle A_n, v, u, t\rangle$ to the two result-

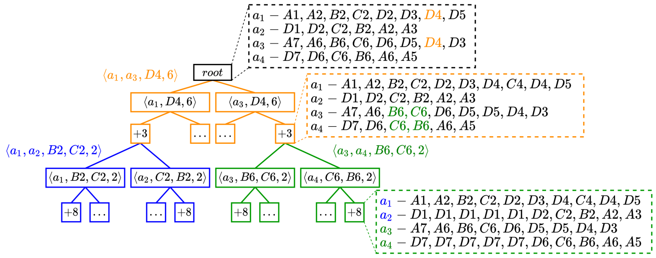


Figure 4: A CT for solving the MAPF instance in Figure 1 with MA-CBS.

ing child CT nodes, respectively. Since both CBS (for resolving external conflicts of two meta-agents) and the joint-state-space MAPF solver (for resolving internal conflicts of a meta-agent) are complete and optimal, MA-CBS is also complete and optimal. The advantage of MA-CBS over CBS is that the internal conflicts of each meta-agent are resolved by merging instead of branching, which avoids (part of) the repeated replanning problem and reduces the size of the CT. For our example from Figure 1, if $b = 3$ and agents a_1 and a_2 are merged during the search of the left blue subtree, then the blue subtrees below the right green subtree will not be generated since all internal conflicts between agents a_1 and a_2 have been resolved. Similarly, agents a_3 and a_4 are merged during the search of the right green subtree, and thus the green subtrees below the left blue subtree will not be generated. The resulting CT is shown in Figure 4, where agents in the same meta-agent are in the same color.

4 Meta-Agent ECBS (MA-ECBS)

Similar to combining CBS with the merging technique to obtain an efficient optimal MAPF solver, we can also combine ECBS with the merging technique to obtain an efficient bounded-suboptimal MAPF solver, which we call MA-ECBS. As long as the joint-state-space MAPF solver for resolving the internal conflicts of meta-agents is complete and optimal, MA-ECBS provides the same suboptimality guarantee as ECBS, meaning that the SoCs of its solutions continue to be at most w times the optimal SoCs.

5 Nested ECBS (NECBS)

Since MA-ECBS finds only bounded-suboptimal solutions, it is unnecessary to use an optimal joint-state-space MAPF solver. For example, one can use ECBS instead of an optimal joint-state-space MAPF solver to improve efficiency, resulting in Nested ECBS (NECBS). For convenience, we call the ECBS that solves the whole MAPF instance the *Outer* ECBS, and the ECBS that resolves internal conflicts of meta-agents the *Inner* ECBS. Suppose NECBS is going to replan a meta-agent A_m in a CT node N^O generated by Outer ECBS, Inner ECBS resolves the internal conflicts of A_m while satisfying the constraints of CT node N^O. Then, it passes the resulting conflict-free paths and the lower bound $N^O.LB^I[A_m]$ to Outer ECBS. We use the smallest lower bound of the CT nodes in the open list OPEN^I of Inner ECBS when it terminates:

$$N^O.LB^I[A_m] = min(N^I.LB | N^I \in \text{OPEN}^I). \tag{5}$$

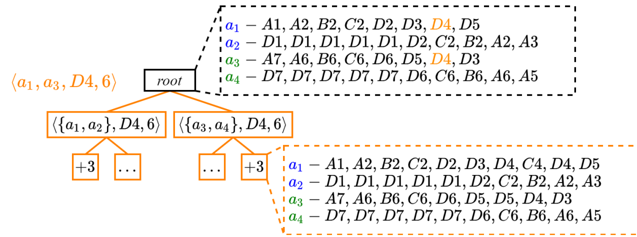


Figure 5: A CT for solving the MAPF instance in Figure 1 with MA-CBS with MR after merging agents a_1 and a_2 into meta-agent $\{a_1, a_2\}$ and merging agents a_3 and a_4 into meta-agent $\{a_3, a_4\}$.

Given M meta-agents $A_1, A_2, \ldots, A_M$ whose sizes are all greater than 1, we define the set of agents that are not in any of these meta-agents as $A_s = \{a_j \mid a_j \notin A_1 \cup A_2 \cup \ldots \cup A_M, 1 \leq j \leq k\}$. We use the same user-specified suboptimality factor w for both Outer ECBS and Inner ECBS. Then, Inner ECBS guarantees that the SoC of the paths of meta-agent A_m is bounded-suboptimal, that is,

$$\sum_{a_i \in A_m} |N^O.paths[i]| \leq w \cdot N^O.LB^I[A_m]. \tag{6}$$

The low-level focal search of Outer ECBS guarantees that the cost of the path of any agent $a_j \in A_s$ is also bounded-suboptimal, meaning that

$$|N^O.paths[j]| \leq \ w \cdot N^O.lb^O[j], \tag{7}$$

where $N^O.lb^O[i]$ is the lower bound of agent a_i at CT node N^O of Outer ECBS. Thus, the cost of CT node N^O of Outer ECBS satisfies the inequality:

$$\begin{aligned} N^O.cost &\leq w \sum_{m=1}^{M} N^O.LB^I[A_m] + w \sum_{a_j \in A_s} N^O.lb^O[j] \\ &= w \cdot N^O.LB^O, \end{aligned} \tag{8}$$

where $N^O.LB^O = \sum_{m=1}^{M} N^O.LB^I[A_m] + \sum_{a_j \in A_s} N^O.lb^O[j]$ is the lower bound of CT node N^O.

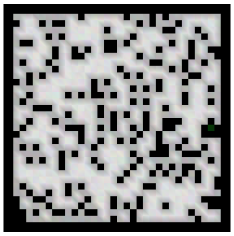
Since Outer ECBS always selects CT nodes from its focal list FOCAL^O, the SoC of its solution is thus bounded by $w \cdot LB^O$, where $LB^O = \min(N^O.LB^O \mid N^O \in \text{OPEN}^O)$ and OPEN^O is the open list of Outer ECBS. Also, the lower-bound improvement of NECBS is based on Outer ECBS, meaning that

$$LB\ Improvement = \hat{LB}^O - N_R^O.LB^O, \tag{9}$$

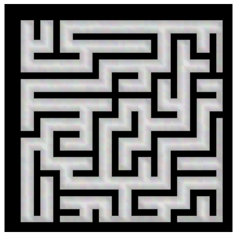
where N_R^O is the root CT node of Outer ECBS, $N_R^O.LB^O$ is the lower bound LB in the beginning of its search, and $\hat{LB}^O$ is the LB^O value once a solution has been found by NECBS.

6 Restart Techniques

Restart techniques are effective for solving many combinatorial problems [Ruan *et al.*, 2002]. In MAPF, two restart techniques have been explored, namely the Merge and Restart (MR) and Rapid Random Restart (RR) techniques.

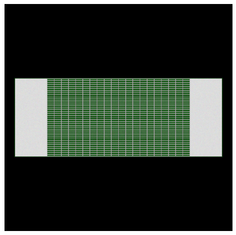


(a) Grid
32×32



(b) Maze
32×32

(c) Den520d
257×256

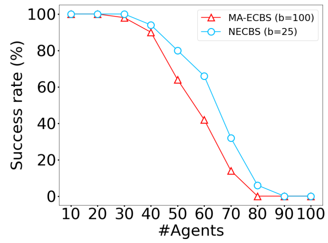


(d) Warehouse
84×170

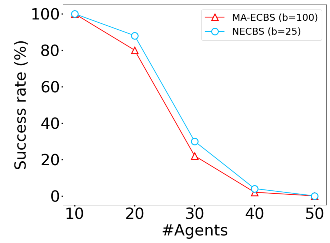
Figure 6: Maps used in the experiments with their sizes given in the form *height* $\times$ *width*.

The Merge and Restart (MR) [Boyarski *et al.*, 2015] technique is an improvement of MA-CBS that further reduces the size of the CT. The MR technique restarts the high-level search at the root CT node after each merging while maintaining all existing meta-agents. When two meta-agents are merged, MA-CBS has already branched $b+1$ times on the external conflicts between the two meta-agents. But with the MR technique, the two meta-agents are already merged after the restart. Thus, the high-level search after the restart never branches on the external conflicts between them, resulting in a smaller CT. For our example, Figure 5 shows the CT after MA-CBS has merged agents a_1 and a_2 into one meta-agent, merged agents a_3 and a_4 into another meta-agent, and restarted the search. Since the MR technique affects neither the high-level nor the low-level search, we can directly use it in MA-ECBS, resulting in MA-ECBS (MR), and NECBS, resulting in NECBS (MR). In the root CT node of MA-ECBS (MR), we use the joint-state-space MAPF solver to plan for meta-agents with sizes greater than 1. In the root CT node of NECBS (MR), we use Inner ECBS to plan for meta-agents with sizes greater than 1. We also propose ECBS with the restart (R) technique, called ECBS (R). Once the number of conflicts exceeds merge threshold b, ECBS (R) shuffles the order of the agents and restarts the search (without merging meta-agents). The order of the agents is important for finding their paths in the root CT node. The agents planned later in the order tend to take bounded-suboptimal detours in order to minimize the number of conflicts with the paths of the agents that have been planned earlier, which affects the number of conflicts in the root CT node and thus the runtime of the search.

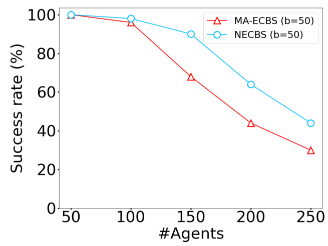
The Rapid Randomized Restart (RR) [Cohen *et al.*, 2018] technique, given the user-specified number of runs #Runs and the runtime limit T (in seconds), restarts the search every



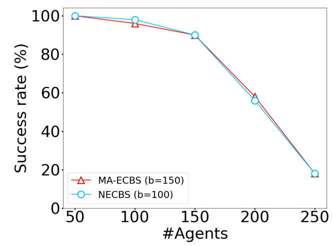
(a) Grid, $w = 1.05$



(b) Maze, $w = 1.05$

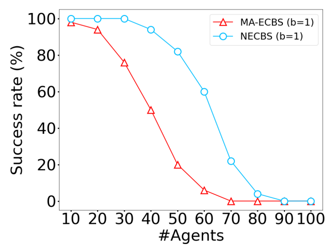


(c) Den520d, $w = 1.01$

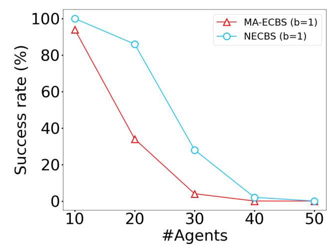


(d) Warehouse, $w = 1.01$

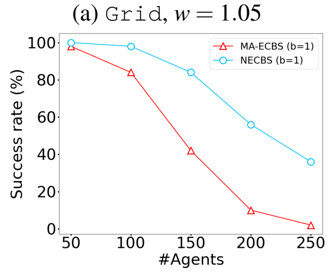
Figure 7: Success rates of MA-ECBS and NECBS for the best merge thresholds b on different maps.



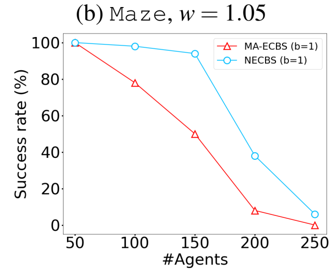
(a) Grid, $w = 1.05$



(b) Maze, $w = 1.05$



(c) Den520d, $w = 1.01$



(d) Warehouse, $w = 1.01$

Figure 8: Success rates of MA-ECBS and NECBS for merge threshold $b = 1$ on different maps.

T/#Runs seconds, each time shuffling the order of the agents before the restart. We denote ECBS with the Rapid Randomized Restart technique as ECBS (RR). It is a state-of-the-art bounded-suboptimal MAPF solver.

7 Empirical Evaluation

We compare our MA-ECBS and NECBS with other ECBS-based MAPF solvers. As shown in Figure 6, the maps used in our experiments are all 4-neighbor grids from the MAPF benchmark suite [Stern *et al.*, 2019], including a 32×32 grid map with 20% blocked cells (Grid), a 32×32 grid map of a maze with alley width 2 (Maze), a 257×256 grid map from

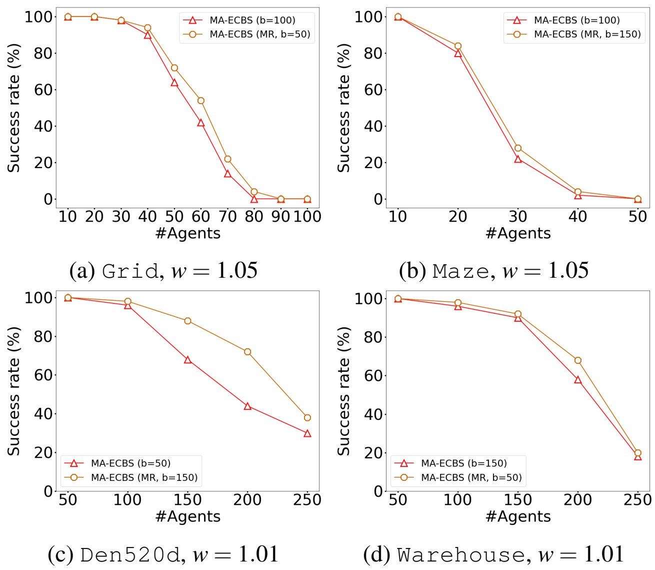
Maps (#Instances, w)	ECBS	#Runs	ECBS (RR)	b	ECBS (R)	MA-ECBS	MA-ECBS (MR)	NECBS	NECBS (MR)
Grid (500, $w = 1.05$)	52.8	5	59.2	1	43.8	34.4	30.4	56.2	49.4
		10	59.8	25	**65.2**	50	53	**57.8**	67
		20	61.2	50	62.4	50.6	**54.4**	56.2	65.8
		30	**62.8**	100	64.4	**50.8**	53.4	55.8	**69.2**
		40	62.6	150	64	50.6	53.4	55.6	69
Maze (250, $w = 1.05$)	42.8	5	47.6	1	38	26.4	25.2	43.2	41.6
		10	48.8	25	48	38.4	38.8	**44.4**	52
		20	**50**	50	**50**	38.8	40.4	43.6	**53.2**
		30	48.8	100	49.6	**40.8**	41.2	44	52
		40	49.6	150	49.6	40.4	**43.2**	**44.4**	52
Den520d (250, $w = 1.01$)	77.2	5	**70.8**	1	54	47.2	44.4	74.8	46.8
		10	51.2	25	71.2	66.4	78.4	78.8	43.4
		20	50.8	50	**77.2**	**67.6**	78.4	**79.2**	**85.6**
		30	56.4	100	75.2	**67.6**	77.6	78.4	84.8
		40	67.2	150	74.4	**67.6**	**79.2**	78.4	84.4
Warehouse (250, $w = 1.01$)	73.6	5	60.4	1	43.6	47.2	42	67.2	47.6
		10	49.6	25	67.6	70.8	75.2	71.2	**80.4**
		20	42	50	68.8	70.8	**75.6**	70.4	76.4
		30	54	100	70	72	73.6	**72.4**	78
		40	**62**	150	**70.4**	**72.4**	73.6	72	74
Overall (1250)	59.8		61.7		65.6	56.5	61.4	62.3	**71.8**

Table 1: Success rates (in percentages) of ECBS, ECBS (RR) with #Runs $\in \{5,10,20,30,40\}$, ECBS (R), MA-ECBS, MA-ECBS (MR), NECBS, and NECBS (MR) with merge thresholds $b \in \{1,25,50,100,150\}$. The number of agents for the Grid map ranges from 10 to 100 in increments of 10. The number of agents for the Maze map ranges from 10 to 50 in increments of 10. The number of agents for the Den520d and Warehouse maps ranges from 50 to 250 in increments of 50. The "Overall" in the bottom row is the average of the highest success rates over the four maps, weighted by their number of MAPF instances. NECBS (MR) solves the most MAPF instances.

the video game Dragon Age: Origin (DAO) (Den520d), and an 84×170 grid map of an automated warehouse (Warehouse). We use both the "even" and "random" scenarios from the benchmark, which yield 50 MAPF instances for each map and each number of agents. The costs between pairs of start and goal vertices are distributed evenly in the MAPF instances of the "even" scenarios. The start and goal vertices are distributed randomly in the MAPF instances of the "random" scenarios. Our main comparison metric is the *success rate*, which is the percentage of the MAPF instances solved within a runtime limit of 5 minutes. For the Grid and Maze maps, which have small sizes but high obstacle densities, we set the suboptimality factor to $w = 1.05$. For the remaining two larger maps with more free space, $w = 1.05$ is too large, resulting in all MAPF solvers having high success rates and preventing us from distinguishing among them. For the Den20d and Warehouse maps, we thus set $w = 1.01$. We implement our MAPF solvers in C++ and run them on servers with 2.80 GHz Intel Xeon Processors E5-2640 v4 and 2 GB RAM. Table 1 shows the success rates of ECBS, ECBS (RR), ECBS (R), MA-ECBS, MA-ECBS (MR), NECBS, and NECBS (MR) with different parameter settings. We discuss these results in detail in the following sections.

7.1 Evaluation of the Merge Thresholds

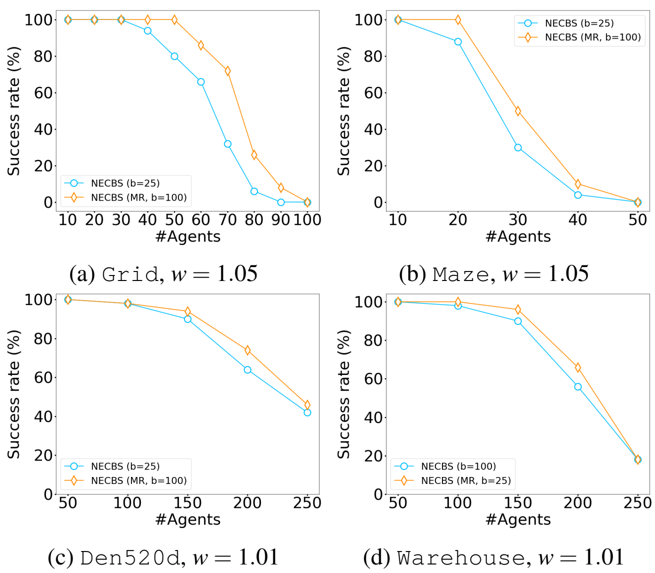
We use a set of merge thresholds $\{1,25,50,100,150\}$ and define the *best merge threshold* of a MAPF solver as the merge threshold in the set that leads to the highest success rate. To demonstrate the efficiency resulting from using different MAPF solvers to resolve internal conflicts of meta-agents, Figure 7 shows the success rates of MA-ECBS and NECBS for their best merge thresholds. The results show that using a joint-state-space MAPF solver like EPEA* (as used by MA-ECBS) instead of Inner ECBS (as used by NECBS) for resolving internal conflicts is more time-consuming on small maps with dense obstacles and thus results in smaller success rates. When the merge threshold b is low, agents are merged frequently, meaning that MAPF solvers are more frequently used to resolve internal conflicts. Thus, the success rates of MA-ECBS are much lower than the ones of NECBS when $b = 1$, as shown in Figure 8.



(a) Grid, $w = 1.05$ (b) Maze, $w = 1.05$

(c) Den520d, $w = 1.01$ (d) Warehouse, $w = 1.01$

Figure 9: Success rates of MA-ECBS and MA-ECBS (MR) for the best merge thresholds b on different maps.



(a) Grid, $w = 1.05$ (b) Maze, $w = 1.05$

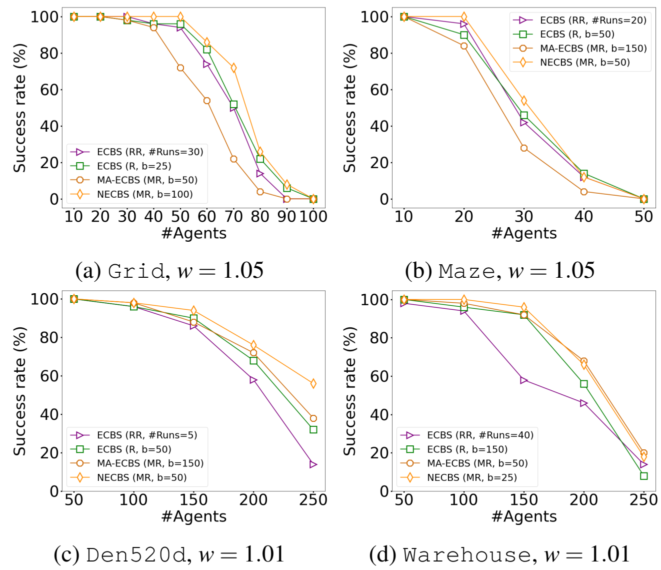
(c) Den520d, $w = 1.01$ (d) Warehouse, $w = 1.01$

Figure 10: Success rates of NECBS and NECBS (MR) for the best merge thresholds b on different maps.

7.2 Evaluation of the Restart Techniques

To demonstrate the efficiency resulting from the MR technique, we compare the success rates of MA-ECBS and MA-ECBS (MR) and the success rates of NECBS and NECBS (MR). Figure 9 shows the success rates of MA-ECBS and MA-ECBS (MR) for their best merge thresholds. Figure 10 shows the success rates of NECBS and NECBS (MR) for their best merge thresholds. The results show that MAPF solvers with the MR technique outperform those without the MR technique.

Figure 11 shows the success rates of MAPF solvers with different restart techniques, namely the RR, R, and MR techniques. These restart techniques result in ECBS (RR), ECBS (R), MA-ECBS (MR), and NECBS (MR). The results show that NECBS (MR) has higher success rates than the MAPF solvers with other restart techniques.



(a) Grid, $w = 1.05$ (b) Maze, $w = 1.05$

(c) Den520d, $w = 1.01$ (d) Warehouse, $w = 1.01$

Figure 11: Success rates of MAPF solvers with different restart techniques for the best merge thresholds b on different maps.

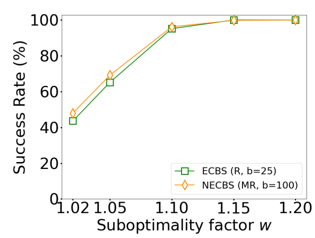
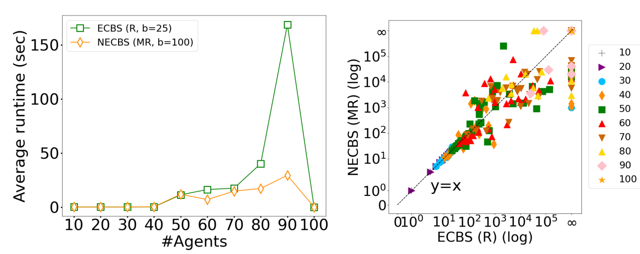
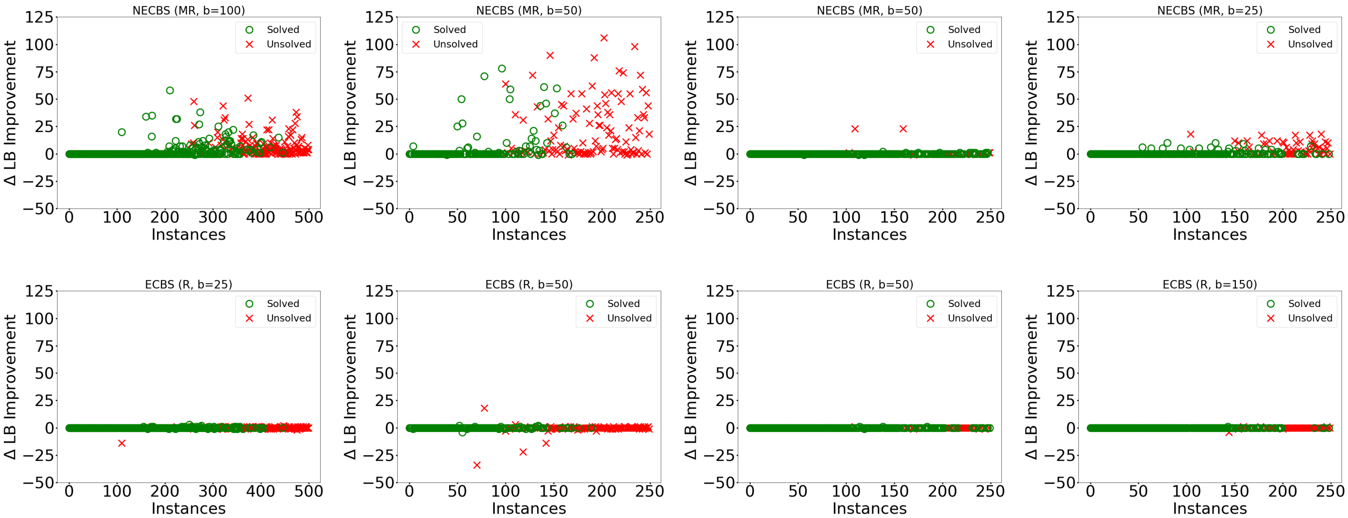


Figure 12: Success rates of NECBS (MR) and ECBS (R) for the best merge thresholds b and increasing suboptimality factor w on the Grid map.



(a) Runtime of NECBS (MR) and ECBS (R). (b) Number of generated CT nodes by NECBS (MR) and ECBS (R).

Figure 13: Results of NECBS (MR) and ECBS (R) for the best merge thresholds b and suboptimality factor $w = 1.05$ on the Grid map. In Subfigure (b), dots of different colors show MAPF instances with different numbers of agents.



(a) `Grid` $w = 1.05$ (b) `Maze` $w = 1.05$ (c) `Den520d` $w = 1.01$ (d) `Warehouse` $w = 1.01$

Figure 14: Δ*LB Improvement*s of NECBS (MR) (top row) and ECBS (R) (bottom row) for the best merge thresholds on different maps. Green circles are the solved MAPF instances, while red crosses are the unsolved ones.

7.3 Evaluation of Merging Agents

Since the difference between the MR and R techniques is whether agents are merged, a comparison between NECBS (MR) and ECBS (R) shows the effect of merging. We use `Grid` MAPF instances where the number of agents ranges from 10 to 100 in increments of 10. Figure 12 shows the success rates of NECBS (MR) and ECBS (R) for their best merge thresholds and suboptimality factors $w \in \{1.02, 1.05, 1.10, 1.15, 1.20\}$. NECBS (MR) has higher success rates than ECBS (R) when the suboptimality factor is small. Both MAPF solvers reach a success rate of 100% once the suboptimality factor increases to 1.2 because, as the suboptimality factor increases, both MAPF solvers have more solutions to choose from. To show that NECBS (MR) outperforms ECBS (R), we analyze the results from the MAPF instances that are solved by both MAPF solvers with their best merge thresholds. Figure 13a shows that NECBS (MR) has a smaller runtime than ECBS (R) on MAPF instances that are solved by both MAPF solvers. Furthermore, we compare the number of CT nodes generated by both MAPF solvers. In Figure 13b, we use a logarithmic scale for both axes. For MAPF instances that are not solved within the runtime limit, we set the number of CT nodes generated by the MAPF solver to infinity. MAPF instances on the right side of the dashed line are the ones that can be solved by NECBS (MR) with fewer generated CT nodes than by ECBS (R). Figure 13b shows that NECBS (MR) solves more MAPF instances with fewer CT nodes than ECBS (R).

Let Δ*LB Improvement* be the difference of *LB Improvement* between a given MAPF solver and ECBS. Figure 14 shows the Δ*LB Improvement*s of NECBS (MR) and ECBS (R) for each MAPF instance. The MAPF instances are sorted in increasing order of their number of agents. The green circles represent the solved MAPF instances, and the red crosses represent the unsolved MAPF instances. As the number of agents increases, NECBS (MR) solves more MAPF instances within the runtime limit and has higher Δ*LB Improvement*s than ECBS (R). One of the reasons is *target symmetry* [Li *et al.*, 2020], which occurs when agent a_i traverses goal vertex g_j of agent a_j after agent a_j has already reached g_j and stayed there. Resolving conflicts between agents a_i and a_j typically results in paths of much higher costs than those of their individual minimum-cost paths (ignoring other agents). The MR technique takes target symmetry into account by merging agents and replanning paths for all agents within a meta-agent with Inner ECBS, which has to increase its lower bound substantially in order to find a solution within the suboptimality bound. In contrast, ECBS (R) simply restarts the search after randomly shuffling the order of the agents, which does not handle target symmetry. Thus, NECBS (MR) solves MAPF instances with higher Δ*LB Improvement*s than ECBS (R).

8 Conclusions

In this paper, we leveraged ideas from two MAPF solvers, namely ECBS and MA-CBS. ECBS finds solutions whose SoCs are guaranteed to be within a user-specified suboptimality factor, and MA-CBS is a variant of optimal CBS that uses meta-agents to handle the repeated replanning problem. We proposed MA-ECBS, which is a variant of ECBS that merges agents into meta-agents and resolves the internal conflicts of meta-agents with a joint-state-space MAPF solver. Furthermore, we proposed NECBS, which is a variant of ECBS that not only merges agents into meta-agents but also resolves the internal conflicts of meta-agents with ECBS. Based on the Merge and Restart (MR) technique, we also proposed the Restart (R) technique, that restarts the search without merging (meta-)agents once the number of conflicts exceeds a given threshold. Our experiments show that NECBS with the MR technique has a higher success rate then the state-of-the-art bounded-suboptimal MAPF solver ECBS with the RR technique.

9 Acknowledgement

The research at the University of Southern California was supported by the National Science Foundation (NSF) under grant numbers 1409987, 1724392, 1817189, 1837779, and 1935712 as well as a gift from Amazon. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the US government.

References

[Barer *et al.*, 2014] Max Barer, Guni Sharon, Roni Stern, and Ariel Felner. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *Proceedings of the 7th Annual Symposium on Combinatorial Search (SoCS)*, pages 19–27, 2014.

[Boyarski *et al.*, 2015] Eli Boyarski, Ariel Felner, Roni Stern, Guni Sharon, David Tolpin, Oded Betzalel, and Solomon Eyal Shimony. ICBS: Improved conflict-based search algorithm for multi-agent pathfinding. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 740–746, 2015.

[Cohen *et al.*, 2018] Liron Cohen, Sven Koenig, T. K. Satish Kumar, Glenn Wagner, Howie Choset, David Chan, and Nathan Sturtevant. Rapid randomized restarts for multi-agent path finding: Preliminary results. In *Proceedings of the 17th International Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, page 1909–1911, 2018.

[Goldenberg *et al.*, 2014] Meir Goldenberg, Ariel Felner, Roni Stern, Guni Sharon, Nathan R. Sturtevant, Robert C. Holte, and Jonathan Schaeffer. Enhanced partial expansion A*. *Journal of Artificial Intelligence Research*, 50:141–187, 2014.

[Hönig *et al.*, 2018] Wolfgang Hönig, James A. Preiss, T. K. Satish Kumar, Gaurav S. Sukhatme, and Nora Ayanian. Trajectory planning for quadrotor swarms. *IEEE Transactions on Robotics*, 34(4):856–869, 2018.

[Li *et al.*, 2020] Jiaoyang Li, Graeme Gange, Daniel Harabor, Peter J. Stuckey, Hang Ma, and Sven Koenig. New techniques for pairwise symmetry breaking in multi-agent path finding. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 193–201, 2020.

[Ma *et al.*, 2017] Hang Ma, Jingxing Yang, Liron Cohen, T. K. Satish Kumar, and Sven Koenig. Feasibility study: Moving non-homogeneous teams in congested video game environments. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 270–272, 2017.

[Morris *et al.*, 2016] Robert Morris, Corina S. Pasareanu, Kasper Søe Luckow, Waqar Malik, Hang Ma, T. K. Satish Kumar, and Sven Koenig. Planning, scheduling and monitoring for airport surface operations. In *AAAI Workshop on Planning for Hybrid Systems*, 2016.

[Pearl and Kim, 1982] Judea Pearl and Jin H. Kim. Studies in semi-admissible heuristics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(4):392–399, 1982.

[Ruan *et al.*, 2002] Yongshao Ruan, Eric Horvitz, and Henry A. Kautz. Restart policies with dependence among runs: A dynamic programming approach. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, page 573–586, 2002.

[Sharon *et al.*, 2012] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Meta-agent conflict-based search for optimal multi-agent path finding. In *Proceedings of the 5th Symposium on Combinatorial Search (SoCS)*, pages 39–40, 2012.

[Sharon *et al.*, 2015] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[Standley, 2010] Trevor Scott Standley. Finding optimal solutions to cooperative pathfinding problems. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, pages 173–178, 2010.

[Stern *et al.*, 2019] Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Roman Barták, and Eli Boyarski. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the 12th International Symposium on Combinatorial Search (SoCS)*, pages 151–159, 2019.

[Veloso *et al.*, 2015] Manuela M. Veloso, Joydeep Biswas, Brian Coltin, and Stephanie Rosenthal. Cobots: Robust symbiotic autonomous mobile service robots. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4423–4429, 2015.

[Wagner and Choset, 2011] G. Wagner and H. Choset. M*: A complete multirobot path planning algorithm with performance bounds. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3260–3267, 2011.