Flex Distribution for Bounded-Suboptimal Multi-Agent Path Finding

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

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

Abstract

Multi-Agent Path Finding (MAPF), the problem of finding collision-free paths with the minimum sum of path costs for multiple agents, is important in many applications, such as automated warehouses. EECBS is a leading two-level algorithm that solves MAPF bounded-suboptimally, that is, within some factor w of the minimum sum of path costs C^*. It uses focal search to find bounded-suboptimal paths on the low level and Explicit Estimation Search (EES) to resolve collisions on the high level. EES keeps track of a lower bound of C^* (LB) and finds paths with sum of path costs at most $w \cdot LB$ in order to guarantee bounded suboptimality. However, the costs of many paths are typically much smaller than w times their individual minimum path costs, meaning that the sum of path costs is much smaller than $w \cdot C^*$. In this paper, we propose Flexible EECBS (FEECBS), which uses a flex(ible) distribution of the path costs to relax the bounded-suboptimality restriction on the low level in order to reduce the number of collisions that need to be resolved on the high level while still guaranteeing to solve MAPF bounded-suboptimally. Flex distribution can make raising LB harder, so we do not only introduce restrictions for using it but also combine it with A* search to avoid the search being slow. Our empirical evaluation shows that the resulting FEECBS speeds up EECBS on MAPF instances with large maps and large numbers of agents and has similar runtimes on MAPF instances with small maps.

Introduction

Multi-Agent Path Finding (MAPF) is the problem of finding collision-free paths for multiple agents, where the set of paths (one for each agent), whose costs are their travel times, is known as the solution. A common objective of MAPF is to minimize the sum of path costs. We say that a solution of MAPF is optimal iff the sum of path costs is minimum, and bounded-suboptimal iff the sum of path costs is a user-specified suboptimality factor w away from minimum. MAPF has many applications, such as autonomous warehouses (Ma et al. 2017), airports (Li et al. 2019d), video games (Li et al. 2020b), and UAVs (Ho et al. 2019).

Conflict-Based Search (CBS) (Sharon et al. 2015) is one of the leading algorithms that solves MAPF optimally. It is a two-level algorithm that finds paths for agents individually on the low level and then resolves collisions on the high level. Once CBS finds a collision between paths of two agents on the high level, it generates two branches, where each branch forces one agent to find a new path in order to avoid the found collision. Although several techniques have been proposed to speed up CBS significantly (Boyarski et al. 2015a,b, 2020b,a; Felner et al. 2018; Gange, Harabor, and Stuckey 2019; Li et al. 2019a,b,c, 2020a), the fact that finding optimal solutions to MAPF is NP-hard [cite] motivates researchers to develop algorithms that find bounded-suboptimal solutions in order to speed up the search.

Explicit Estimation CBS (EECBS) (Li, Ruml, and Koenig 2021) is a variant of CBS that guarantees to find bounded-suboptimal solutions. Its bounded suboptimality is achieved by replacing A* search with Explicit Estimation Search (EES) (Thayer and Ruml 2011) on the high level and focal search (Pearl and Kim 1982) on the low level of CBS. EES keeps track of a lower bound LB of the sum of path costs of the optimal solution and resolves collisions among the paths of all agents whose sum of path costs is at most $w \cdot LB$. Focal search maintains a lower bound on the individual minimum path cost for each agent and finds a path whose cost is at most w away from the lower bound. In Li, Ruml, and Koenig (2021), the authors also modified the speed-up techniques of CBS so that they can be used to speed up EECBS, namely the prioritizing conflicts, bypassing conflicts, symmetric reasoning, and the WDG heuristic.

We call the difference between the suboptimality bound $w \cdot LB$ and the sum of path costs of the agents the *flex*. In this paper, we propose Flexible EECBS (FEECBS) to distribute the flex among agents, allowing some agents to take paths whose costs are not bounded-suboptimal, as long as the sum of path costs of the agents is bounded-suboptimal. However, FEECBS has the drawback that, in comparison to EECBS, it seldom increases the LB. Thus, we propose restrictions on the flex distribution and even restart the search with EECBS. Since increasing the lower bound of a path may also increase the LB, we propose focal-A* search so that FEECBS can switch to A* search on the low level to increase the lower bound of a path. Our empirical evaluation shows that although FEECBS has similar runtimes with EECBS on small maps, it substantially improves the efficiency of EECBS on MAPF instances with large maps and

large numbers of agents. A preliminary study of this work has been published as (Chan et al. 2021).

Preliminaries

In this section, we introduce MAPF and EECBS as well as speed-up techniques for its search.

Multi-Agent Path Finding (MAPF)

In this paper, we use the MAPF definition in Stern et al. (2019). The MAPF problem consists of an undirected graph and a set of k agents $\{a_1, \ldots, a_k\}$. Here, we assume that the graph is a 4-neighbor grid. Each agent a_i has a unique start vertex s_i and a unique goal vertex g_i. Time is discretized into timesteps. At each timestep, an agent is allowed to either move to an adjacent vertex or wait at its current vertex. After reaching its goal vertex, an agent permanently waits there. A *path* of an agent, starting at start vertex and ending at goal vertex, is a sequence of vertices indicating where the agent is at each timestep. The *cost* of a path is the number of timesteps needed by the agent to move from its start vertex to its goal vertex, ignoring the timesteps when it permanently waits at its goal vertex. Two paths have a *conflict* iff two agents stay at the same vertex or traverse the same edge in opposite directions at the same timestep. A *solution* is a set of conflict-free paths, one for each agent. An *optimal solution* is a solution with the minimum *sum of the costs (SoC)* of the paths, denoted as C^*. A *bounded-suboptimal solution* for a user-specified suboptimality factor $w \geq 1$ is a solution with a SoC of at most $w \cdot C^*$.

Explicit Estimation CBS (EECBS)

EECBS is a two-level algorithm that solves MAPF bounded-suboptimally for a user-specified suboptimality factor w. On the high level, EECBS constructs a *Constraint Tree* (CT). Each CT node N contains a set of constraints $constraints(N)$ that coordinate agents to avoid conflicts. For each agent a_i, CT node N contains a path that satisfies $constraints(N)$ with cost $c_i(N)$. Let $c_i^*(N)$ be the minimum cost of the paths that satisfy $constraints(N)$ and $lb_i(N)$ be the lower bound of $c_i^*(N)$. Since EECBS uses focal search on the low level, the path of each agent is guaranteed to be bounded-suboptimal, meaning that $c_i(N) \leq w \cdot c_i^*(N)$. Let $c(N)$ and $c^*(N)$ denote $\sum_{i=1}^k c_i(N)$ and $\sum_{i=1}^k c_i^*(N)$, respectively. The root CT node contains no constraints. Given a conflict between two paths in CT node N selected for expansion, EECBS resolves it by generating two child CT nodes, each with an additional constraint that prohibits one of the conflicting agents from using the contested vertex or edge at the conflicting timestep.

On the high level, EECBS maintains three lists of CT nodes: CLEANUP$_H$, OPEN$_H$, and FOCAL$_H$. CLEANUP$_H$ is a regular open list of A* search which sorts the CT nodes N in increasing order of an admissible cost function $f(N) = g(N) + h(N)$, where $g(N)$ is $\sum_{i=1}^k lb_i(N)$ and $h(N)$ is an admissible heuristic, which is a cost-to-go function that under-estimates the difference between the cost of the optimal solution in the subtree of CT node N and $g(N)$. OPEN$_H$ is another regular open list of A* search which sorts the CT nodes in increasing order of another cost function $\hat{f}(N) = g(N) + \hat{h}(N)$, where $\hat{h}(N)$ is a more informed but potentially inadmissible heuristic. Let N_f be a CT node in CLEANUP$_H$ with the minimum f-value LB and $N_{\hat{f}}$ be a CT node in OPEN$_H$ with the minimum $\hat{f}$-value. FOCAL$_H$ contains those CT nodes N in OPEN$_H$ with $\hat{f}(N) \leq w \cdot \hat{f}(N_{\hat{f}})$, sorted in increasing order of the distance-to-go function $h_c(N)$, which is the number of conflicts in the set of paths of CT node N. Let N_{h_c} be the CT node in FOCAL$_H$ with the minimum h_c-value. At each iteration, EECBS uses the following rules to select a CT node for expansion:

(E1) If $c(N_{h_c}) \leq w \cdot f(N_f)$, then expand N_{h_c}.

(E2) Else if $c(N_{\hat{f}}) \leq w \cdot f(N_f)$, then expand $N_{\hat{f}}$.

(E3) Else, expand N_f.

Thus, each expanded CT node N satisfies

$$c(N) = \sum_{i=1}^{k} c_i(N) \leq w \cdot f(N_f) \leq w \cdot C^*, \qquad (1)$$

where $w \cdot f(N_f)$ guarantees the bounded suboptimality of the solution since $f(N_f) = LB \leq C^*$. The smaller the number of conflicts among the paths of the CT node is, the more likely EECBS is to find a solution with fewer CT node expansions in the subtree of this CT node. The smaller the $\hat{f}$-value of the CT node is, the more likely EECBS is to find a solution with a smaller cost in the subtree of this CT node. Thus, EECBS uses Rules (E1) or (E2) to speed up the search by expanding CT nodes with the minimum number of conflicts or the minimum $\hat{f}$-value. EECBS uses Rule (E3) to increase LB such that EECBS is more likely to expand CT nodes with Rules (E1) or (E2) in the future. To compute the admissible heuristic efficiently, if a CT node N is either the root CT node or expanded from CLEANUP$_H$, then EECBS uses the WDG heuristic (Li et al. 2019a) (which is informed but time-consuming to compute). Otherwise, it uses the pathmax heuristic (which is less informed but faster to compute), defined as $f(\hat{N}) - g(N)$, where $\hat{N}$ is the parent CT node of N.

On the low level, given a CT node N, EECBS uses vertex-timestep nodes $n = (v, t)$ or, for short, v-t nodes to represent the case that an agent a_i stays at vertex v at timestep t. It performs focal search to find a bounded-suboptimal path for the agent and a *low-level lower bound* $lb_i(N)$ on the minimum cost of its paths, where the paths have to satisfy all constraints in $constraints(N)$. The low-level search maintains two lists of v-t nodes: OPEN$_L$ and FOCAL$_L$. OPEN$_L$ is the regular open list of A* search and sorts v-t nodes in increasing order of their f-values $f(n)$, which is an admissible cost function. FOCAL$_L$ contains those v-t nodes in OPEN$_L$ with $f(n) \leq \tau_i(N) = w \cdot f_{min,i}$, where $f_{min,i}$ is the minimum f-value of all nodes n in OPEN$_L$ and $\tau_i(N)$ is the *low-level focal threshold*. That is, $f_{min,i}$ is guaranteed to be a low-level lower bound on the cost $c_i^*(N)$ of the optimal path that satisfies $constraints(N)$, and $\tau_i(N)$ guarantees that the found path is bounded-suboptimal. FOCAL$_L$ sorts these v-t nodes in increasing order of the number of conflicts $d(n)$

with the paths of the other agents when agent a_i moves from start vertex s_i at timestep 0 to vertex v at timestep t. The low-level search first updates FOCAL$_L$ and $f_{min,i}$, if necessary, and then expands the v-t node n with the minimum d-value. Since the f-values of the v-t nodes in FOCAL$_L$ are at most $\tau_i(N)$, EECBS always finds a bounded-suboptimal path whose cost $c_i(N)$ satisfies

$$f_{min,i}(N) \leq c_i(N) \leq \tau_i(N) = w \cdot f_{min,i}(N), \quad \forall i \in \{1, 2, \ldots, k\}. \tag{2}$$

EECBS updates $f_{min,i}(N)$ during the low-level search and sets $lb_i(N)$ to $f_{min,i}(N)$ after the low-level search terminates. Then,

$$lb_i(N) \leq c_i(N) \leq w \cdot lb_i(N) \leq w \cdot c_i^*(N), \tag{3}$$

where $w \cdot lb_i(N)$ is the *low-level suboptimality bound*. Thus, the cost of CT node N satisfies

$$c(N) = \sum_{i=1}^{k} c_i(N) \leq w \cdot \sum_{i=1}^{k} lb_i(N) = w \cdot g(N). \tag{4}$$

Prioritizing Conflicts for EECBS

Prioritizing conflicts (Boyarski et al. 2015b) is a conflict-selection technique that is originally used in CBS. Suppose that EECBS resolves a conflict by expanding CT node $\hat{N}$ and generating two child CT nodes N and N'. The conflict is *cardinal* iff $c^*(N) > c^*(\hat{N})$ and $c^*(N') > c^*(\hat{N})$. It is *semi-cardinal* iff either $c^*(N) > c^*(\hat{N})$ or $c^*(N') > c^*(\hat{N})$ holds, and *non-cardinal* iff $c^*(N) \leq c^*(\hat{N})$ and $c^*(N') \leq c^*(\hat{N})$. By resolving cardinal conflicts, EECBS tends to raise its LB, and thus prioritizing conflicts is likely to speed up EECBS.

Bypassing Conflicts for EECBS

Bypassing conflicts (Boyarski et al. 2015a) is a technique originally used in CBS. Instead of keeping the child CT nodes after expansion, it replaces the paths of the current CT node with the paths of one of the child CT nodes if the paths of that child CT node have the same cost as the corresponding paths of the current CT node but fewer conflicts. However, since paths are bounded-suboptimal, EECBS can relax the condition on their costs. While expanding CT node $\hat{N}$ and generating its child CT node N, the high-level search bypasses conflicts if (C1) CT node $\hat{N}$ is not from CLEANUP$_H$, (C2) $c_i(N) \leq w \cdot lb_i(\hat{N})\ \forall i \in \{1, 2, \ldots, k\}$, (C3) $c(N) \leq w \cdot f(N_f)$, and (C4) $h_c(N) < h_c(\hat{N})$. EECBS uses (C1) to speed up the search, as CT nodes expanded from CLEANUP$_H$ are typically used to increase LB (as shown in (E3)), and thus it is not necessary to bypass their conflicts. It uses (C2) and (C3) to guarantee the bounded suboptimality. It uses (C4) to reduce the number of conflicts and avoid repeatedly finding the same set of paths while bypassing conflicts. Since $lb_i(N)$ uses $constraints(N)$ that contains more constraints than $constraints(\hat{N})$, EECBS does not replace $lb_i(\hat{N})$ with $lb_i(N)$ while bypassing conflicts.

Flexible EECBS (FEECBS)

EECBS uses focal search on the low level, thus each path is guaranteed to be bounded-suboptimal (see Inequality (3)). However, since EECBS finds a bounded-suboptimal solution by only expanding CT nodes that satisfy Inequality (1) on the high level, it is not necessary to insist that the cost of the path of each agent is at most its suboptimality bound. Thus, we propose Flexible EECBS (FEECBS), which relaxes the bounded suboptimality of each path while guaranteeing the bounded suboptimality of the solution, meaning that Inequality (4) still holds for every CT node. By distributing the flex among the agents, FEECBS can reduce $h_c(N)$ and relax the constraints of bypassing conflicts.

Resolving Conflicts with Flex Distribution

If FEECBS sequentially replans multiple paths of the agents in a CT node, such as when it resolves a target conflict (Li et al. 2020a) where an agent on its goal vertex blocks other agents, then FEECBS has to update the suboptimality bound of the replanned path before replanning the path of the next agent. In the following, we focus on the situation where FEECBS replans only one path in a CT node. Suppose that EECBS expands CT node $\hat{N}$ and generates one of its child CT nodes N. We first define the *flex* (over all k agents) of CT node N as

$$\begin{aligned} \Delta(N) &= w \cdot g(N) - c(N) \\ &= w \cdot \sum_{j=1}^{k} lb_j(N) - \sum_{j=1}^{k} c_j(N) \\ &\geq 0 \text{ (due to Inequality (4)).} \end{aligned} \tag{5}$$

While replanning the path of agent a_i in CT node N, we define the flex over the other $k-1$ agents that excludes agent a_i as

$$\Delta_i(N) = w \cdot \sum_{j=1, j\neq i}^{k} lb_j(N) - \sum_{j=1, j\neq i}^{k} c_j(N). \tag{6}$$

The low-level search now finds a path that satisfies

$$lb_i(N) \leq c_i(N) \leq w \cdot lb_i(N) + \Delta_i(N). \tag{7}$$

Intuitively, if $\Delta_i(N)$ is positive, meaning that there exists at least one agent $a_{j\neq i}$ whose $c_j(N)$ is less than $w \cdot lb_j(N)$, then we can increase $\tau_i(N)$ with the flex over the other agents. By using flex during the low-level search, we can relax the $\tau_i(N)$ from $w \cdot f_{min,i}$ to

$$\tau_i(N) = w \cdot \max\{f_{min,i}, lb_i(\hat{N})\} + \Delta_i(N). \tag{8}$$

The term $\max\{f_{min,i}, lb_i(\hat{N})\}$ guarantees the relaxed $\tau_i(N)$ to be at least $lb_i(\hat{N})$ and thus FOCAL$_L$ to be non-empty during the low-level search.[1] More specifically,

[1] We use $g(N)$ instead of $f(N)$ to define the flex because, otherwise, we cannot compute the flex over the other $k-1$ agents in Equation (6). The reason is that EECBS uses the WDG and path-max heuristics to estimate the difference between the SoC of the optimal solution in the subtree of CT node N and $g(N)$, and thus it does not know the difference between the individual minimum cost of the paths of agent a_i and $c_i(N)$.

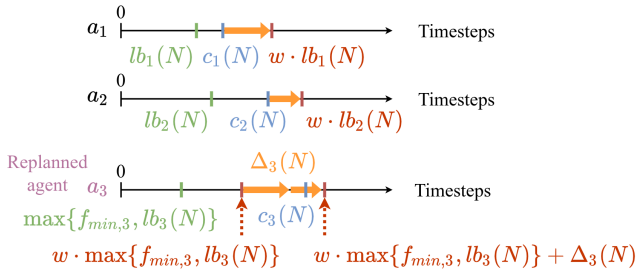


Figure 1: An illustrative example how positive flex increases the low-level focal threshold. Suppose FEECBS generates CT node N and replans the path of agent a_3 with positive flex over agents a_1 and a_2. The green bars are the low-level lower bounds, the blue bars are the path costs, and the red bars are the low-level focal thresholds.

FEECBS may find paths for some agents $a_{j\neq i}$ whose $c_j(N)$ is larger than $w \cdot lb_j(N)$, which results in $\Delta_i(N) < 0$. However, according to Inequality (5), $\Delta(N)$ should be non-negative. Thus, $\Delta_i(N)$ should satisfy

$$\begin{aligned}\Delta_i(N) &= w \cdot \sum_{j=1,j\neq i}^{k} lb_j(N) - \sum_{j=1,j\neq i}^{k} c_j(N) \\ &= w \cdot \sum_{j=1,j\neq i}^{k} lb_j(\hat{N}) - \sum_{j=1,j\neq i}^{k} c_j(\hat{N}) \\ &= \Delta(\hat{N}) - w \cdot lb_i(\hat{N}) + c_i(\hat{N}) \\ &\geq -w \cdot lb_i(\hat{N}) + lb_i(\hat{N}).\end{aligned} \tag{9}$$

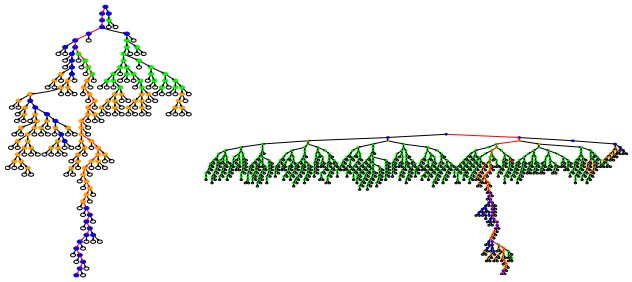
By using the term $\max\{f_{min,i}(N), lb_i(\hat{N})\}$, $\tau_i(N)$ satisfies

$$\begin{aligned}&w \cdot \max\{f_{min,i}(N), lb_i(\hat{N})\} + \Delta_i(N) \\ &\geq w \cdot \max\{f_{min,i}(N), lb_i(\hat{N})\} - w \cdot lb_i(\hat{N}) + lb_i(\hat{N}) \\ &= w \cdot \max\{f_{min,i}(N) - lb_i(\hat{N}), 0\} + lb_i(\hat{N}) \\ &\geq \max\{f_{min,i}(N) - lb_i(\hat{N}), 0\} + lb_i(\hat{N}) \\ &\geq f_{min,i}(N).\end{aligned} \tag{10}$$

Thus, $\tau_i(N) \geq f_{min,i}(N)$ holds, which guarantees that FOCAL$_L$ is always non-empty and the low-level search is thus able to find a bounded-suboptimal path for agent a_i. Also, the low-level search becomes an A* search when all flex has been distributed to agents other than agent a_i in the ancestor CT nodes of CT node N. Thus, FEECBS is still complete and bounded-suboptimal. Figure 1 shows an illustrative example of the variable relations when using positive flex to increase the low-level focal threshold of the replanned agent.

Bypassing Conflicts with Flex Distribution

When EECBS expands CT node $\hat{N}$ and generates its child CT node N, Condition (C2) guarantees that the cost of each replaced path is still within its low-level suboptimal bound of CT node $\hat{N}$. For FEECBS, since we relax the suboptimal bound of the path of each agent a_i by using flex distribution,



(a) CT of EECBS (b) CT of FEECBS

Figure 2: The CTs of EECBS and FEECBS. The blue, orange, and greed circles are the CT nodes selected by Rules (E1), (E2), and (E3) for expansion respectively. The white circles are the CT nodes that have not been expanded. The lines are the edges connecting the parent and child CT nodes, where the red lines indicate a path in CT that finds a solution.

	Rule (E1)	Rule (E2)	Rule (E3)
EECBS	30	72	31
FEECBS	429	51	36

Table 1: The numbers of CT nodes selected by Rules (E1), (E2), and (E3) for expansion of Figure 2.

the path cost does not need to satisfy $c_i(N) \leq w \cdot lb_i(N)$ as long as $\Delta(N) \geq 0$ (see Inequality (5)). Thus, to bypass conflicts with flex distribution, we maintain Conditions (C1), (C3), and (C4) and relax Condition (C2) to

$$c(N) = \sum_{i=1}^{k} c_i(N) \leq w \cdot g(\hat{N}) = w \cdot \sum_{i=1}^{k} lb_i(\hat{N}). \tag{11}$$

We compare FEECBS with and without bypassing conflicts over 7000 MAPF instances on 8 different maps using the setup described in the Empirical Evaluation Section. We add to both algorithms all the other speed-up techniques, namely prioritizing conflicts, symmetric reasoning, and using the WDG heuristic. The result shows that by using bypassing conflicts, FEECBS can solve 50 more MAPF instances.

Limitations of Flex Distribution

When finding a path of agent a_i in CT node N whose parent CT node is $\hat{N}$, FEECBS reduces the number of conflicts by relaxing $\tau_i(N)$. However, the relaxed $\tau_i(N)$ has two limitations. Suppose $w \cdot lb_i(\hat{N}) < c_i^*(N) \leq w \cdot lb_i(\hat{N}) + \Delta_i(N)$. For EECBS, we have

$$w \cdot lb_i(\hat{N}) < c_i^*(N) \leq c_i(N) \leq w \cdot lb_i(N), \tag{12}$$

which results in $lb_i(\hat{N}) < lb_i(N)$. For FEECBS, it is possible to find a path with cost $c_i(N) \leq w \cdot lb_i(\hat{N}) + \Delta_i(N)$, in which case $lb_i(\hat{N})$ may be equal to $lb_i(N)$ and thus results in poor LB improvement. On the other hand, since FEECBS uses flex, $c(N)$ may be larger than that of EECBS. Both the poor LB improvement and the large $c(N)$ may result

in fewer CT nodes in FEECBS that satisfy $c(N) \leq w \cdot LB$. Thus, the first limitation (L1) is that FEECBS tends to select CT nodes by Rule (E3) for expansion more frequently than EECBS. Figure 2 and Table 1 show the CTs and the numbers of CT nodes selected for expansion of EECBS and FEECBS with $w = 1.02$ and all the speed-up techniques while solving the same MAPF instance that has a maze-like grid map (the maze map in the empirical evaluation section) with 40 agents, where FEECBS selects more CT nodes by Rule (E3) for expansion. If all the paths satisfying $constraints(N)$ contain at least m conflicts, then both EECBS and FEECBS have to expand all v-t nodes n in FOCAL$_L$ that satisfy $d(n) < m$. Since FEECBS typically contains more v-t nodes in FOCAL$_L$ than EECBS due to the relaxed $\tau_i(N)$, the second limitation (L2) is that FEECBS is likely to expand more v-t nodes n that satisfy $d(n) < m$.

To overcome the limitation (L1), we add flex restrictions to FEECBS such that it does not always distribute flex on the agents. Also, we restart the search with EECBS if the number of CT nodes selected by Rule (E3) for expansion exceeds a user-specified limit T_N. To overcome limitation (L2), we use low-level focal-A* search that switches from focal search to A* search if the number of expanded v-t nodes exceeds another user-specified limit $T_i(N)$. We will give the details in the next two sections.

Flex Restrictions and Restart with EECBS

When finding a path of agent a_i in CT node N whose parent CT node is $\hat{N}$, if $\Delta_i(\hat{N}) < 0$, then we should always use flex distribution in CT node N in order to satisfy Inequality (4). Otherwise, we set up restrictions on using flex distribution to increase $lb_i(N)$ from $lb_i(\hat{N})$. That is, we prohibit FEECBS to distribute the flex and use the original focal search if at least one of the following restriction conditions holds: (R1) CT node $\hat{N}$ is the root CT node, (R2) CT node $\hat{N}$ is selected by Rule (E3) for expansion, or (R3) The resolved conflict is cardinal. We use Restriction (R1) to avoid that the low-level search distributes the entire flex to one agent in the beginning of the search. We use Restrictions (R2) and (R3) since both selecting CT nodes by Rule (E3) for expansion and resolving cardinal conflicts are likely to increase $lb_i(N)$ from $lb_i(\hat{N})$.

Although we set up restrictions on using flex distribution, EECBS can still be faster than FEECBS in increasing the LB. Thus, when using FEECBS, we will restart the search with EECBS if the number of CT nodes that are continuously selected by Rule (E3) for expansion exceeds a user-specified CT node limit T_N.

Low-Level Focal-A* Search

To alleviate the limitations of FEECBS, we switch from the low-level focal search to the low-level A* search if the number of generated v-t nodes exceeds the v-t node limit $T_i(N)$. That is, we select v-t nodes in OPEN$_L$ instead of FOCAL$_L$ for expansion if the number of conflicts no longer leads to finding a bounded-suboptimal path. By switching from focal search to A* search, we alleviate Limitation (L1) since A* search tends to raise the $f_{min,i}(N)$ by selecting the v-t node n with the minimum $f(n)$ for expansion, and Limitation (L2) since we no longer select v-t node in FOCAL$_L$ for expansion. Also, since switching to A* search does not depend on flex, we can also use low-level focal-A* search for EECBS.

Algorithm 1: Low-level focal-A* search

Input: CT node N, parent CT node $\hat{N}$, agent a_i with start vertex s_i and goal vertex g_i

Compute $\Delta_i(N)$ and $T_i(N)$
root v-t node $r \leftarrow (s_i, 0)$
$\eta_i(N) \leftarrow 1$
$f_{min,i}(N) \leftarrow f(r)$
if *use flex restrictions* ***and*** $\Delta_i(N) \geq 0$ **then** $\delta \leftarrow 0$
else $\delta \leftarrow \Delta_i(N)$
$\tau_i(N) \leftarrow w \cdot \max\{f_{min,i}(N), lb_i(\hat{N})\} + \delta$
INSERTNODE(r)
while *OPEN$_L$ is not empty* **do**
  if $\eta_i(N) \leq T_i(N)$ **then**
    $n \leftarrow$ Pop the top v-t node from FOCAL$_L$
    Erase n from OPEN$_L$
  else $n \leftarrow$ Pop the top v-t node from OPEN$_L$
  if ISGOAL(n) **then return** EXTRACTPATH(n)
  $neighbors \leftarrow$ EXPANDNODE(n)
  for $n' \in neighbors$ **do**
    if n' *satisfies* $constraints(N)$ **then**
      $\eta_i(N) \leftarrow \eta_i(N) + 1$
      INSERTNODE(n)
  if $f_{min,i}(N) < f$(top v-t node in OPEN$_L$) **then**
    $f_{min,i}(N) \leftarrow f$(top v-t node in OPEN$_L$)
    if $\eta_i(N) \leq T_i(N)$ **then**
      $\tau_i' \leftarrow w \cdot \max\{f_{min,i}(N), lb_i(\hat{N})\} + \delta$
      UPDATEFOCAL($\tau_i(N), \tau_i'$)
      $\tau_i(N) \leftarrow \tau_i'$
return "No path satisfies $constraints(N)$."

However, determining $T_i(N)$ is instance-dependent, and thus difficult. Here, we provide an intuitive guideline that works empirically. Suppose that FEECBS finds the path of agent a_i at the ancestor CT node $\tilde{N}$ of CT node N and did not replan this path until CT node N. We denote the number of generated v-t nodes of agent a_i in CT node $\tilde{N}$ as $\eta_i(\tilde{N})$. While generating CT node N, FEECBS resolves conflicts by adding constraints, which typically generates more v-t nodes than CT node $\tilde{N}$ in order to find paths that satisfy the constraints. That is, the number of previously generated v-t nodes in CT node $\tilde{N}$ can be used to estimate the v-t nodes needed in the CT node N. We define the v-t node limit, denoted as $T_i(N)$, as

$$T_i(N) = \kappa \cdot \eta_i(\tilde{N}), \tag{13}$$

where κ is a user-specified factor. That is, the v-t node limit increases as the number of the generated v-t nodes in the ancestor CT node increases. Figure 3, for instance, shows the success rates of FEECBS with $w = 1.02$, speed-up tech-

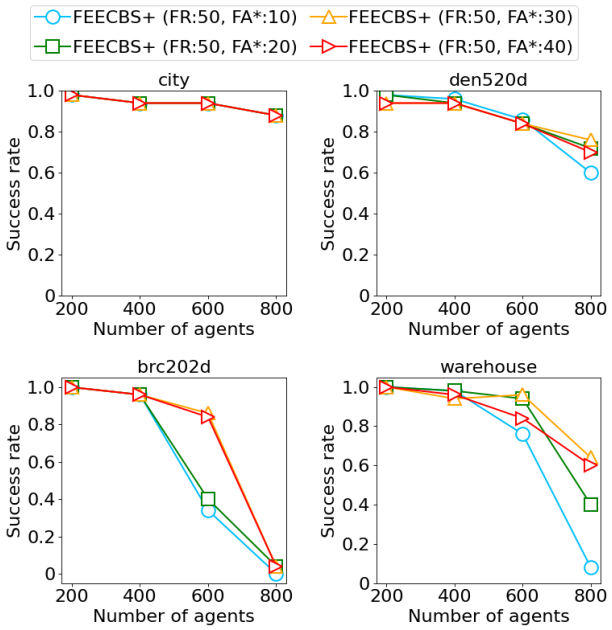


Figure 3: Success rates of FEECBS on 4 large maps (city, den520d, brc202d, and warehouse), where the configurations of the maps are shown in the Empirical Evaluation section.

niques, flex restrictions with $T_N = 50$, and low-level focal-A* search with $T_i = \{10, 20, 30, 40\}$ on 4 large maps.

Algorithm 1 shows the pseudo-code of low-level focal-A* search with flex distribution and flex restrictions. To begin with, we compute $\Delta_i(N)$ and $T_i(N)$ and initialize the root v-t node $(s_i, 0)$, $\eta_i(N)$, and $f_{min,i}(N)$ [Lines 1 to 4]. If the flex restrictions are triggered and the flex over other agents is non-negative, then we ignore $\Delta_i(N)$ [Lines 5 to 7]. If the number of generated v-t nodes is within the v-t node limit, then we run focal search (with or without flex distribution) [Lines 10 to 12] and update FOCAL$_L$ [Line 24]. Otherwise, we switch to A* by expanding v-t nodes from OPEN$_L$ [Line 13] and stop updating FOCAL$_L$. Supposed that $n = (v, t)$. Function INSERTNODE(n) inserts v-t node n into OPEN$_L$. If $f(n) \leq \tau_i(N)$, then it insert v-t node n into FOCAL$_L$. Function ISGOAL(n) checks if vertex v is the goal vertex g_i and if v-t node n satisfies $constraints(N)$. Function EXTRACTPATH(n) generates a path by iteratively backtracking the parent v-t nodes from v-t node n to the root v-t node. Function EXPANDNODE(n) takes all the reachable vertices v' from vertex v (including vertex v) and forms the v-t nodes (v', t'). To terminate the search within the finite runtime limit, timestep t' is set to t if all the other agents has reach their goal vertices before timestep t, and $t' = t + 1$ otherwise. Function UPDATEFOCAL$(\tau_i(N), \tau_i')$ inserts v-t nodes n, whose $\tau_i(N) \leq f(n) \leq \tau_i'(N)$, from OPEN$_L$ into FOCAL$_L$.

Empirical Evaluation

We evaluate the algorithms on 8 maps from the MAPF benchmark suite (Stern et al. 2019). We use 4 large maps, namely a 256 × 256 grid map (`Berlin_1_256`, denoted as "city"), a 256 × 257 grid map and a 530 × 481 grid map from the video game Dragon Age: Origin (DAO) (`den520d` and `brc202d`), and a 321 × 123 grid map from the warehouse scenario (`warehouse-20-40-10-2-1`, denoted as "warehouse"). The number of agents ranges from 200 to 800 in increments of 200. We also use 4 small maps with size 32 × 32, namely grid map `maze-32-32-2` (denoted as "maze") with the number of agents ranging from 10 to 50 in increments of 10, grid map `room-32-32-4` (denoted as "room") with the number of agents ranging from 10 to 50 in increments of 10, grid map `random-32-32-20` (denoted as "random") with the number of agents ranging from 20 to 100 in increments of 20, and grid map `empty-32-32` (denoted as "empty"). with the number of agents ranging from 50 to 200 in increments of 50. We use the "even" and the "random" scenarios, which each yields 25 MAPF instances for each map and number of agents. We use 4 suboptimality factors w, namely 1.01, 1.02, 1.05, and 1.1. The algorithms are implemented in C++, and the experiments are conducted with CentOS Linux on an AMD EPYC 7302 16-Core Processor with memory limit of 16 GB. The main comparison metrics are the runtime and the success rate, which is the percentage of MAPF instances solved within the runtime limit of 60 seconds per MAPF instance.

We evaluate EECBS+ (EECBS with all the speed-up techniques), FEECBS+ (FEECBS with all speed-up techniques), FEECBS+(FR) (FEECBS with all speed-up techniques, flex restrictions with $T_N = 50$, and restart with EECBS), EECBS+(FA*) (EECBS with all speed-up techniques and low-level focal-A* search $\kappa = 20$), and FEECBS+(FR,FA*) (FEECBS with all speed-up techniques, flex restrictions $T_N = 50$, restart with EECBS, and low-level focal-A* search with $\kappa = 20$). Figure 4 shows the average runtimes for different suboptimality factors. Table 2 shows the success rates for suboptimality factors 1.01 and 1.05. On large maps, if the suboptimality factor is small (i.e., $w = 1.01$), then FEECBS+ outperforms EECBS+ since the flex distribution speeds up the search by relaxing the low-level focal threshold. However, as the suboptimality factor increases, FEECBS+ improves less over EECBS+ since due to Limitations (L1) and (L2). On small maps, small improvements on LB can typically speeds up the search. Thus, FEECBS+ is less efficient than EECBS+ due to Limitation (L1). However, with flex restrictions and restart with EECBS, the success rates of FEECBS+(FR) are competitive with the ones of EECBS+ on small maps, and, with low-level focal-A* search, both EECBS+(FA*) and FEECBS+(FR,FA*) show improvements.

Conclusion

We proposed flex distribution to relax the bounded-suboptimality restrictions of EECBS while still finding paths on the low level that result in bounded-suboptimal solutions. We also proposed flex restrictions, restart with EECBS, and

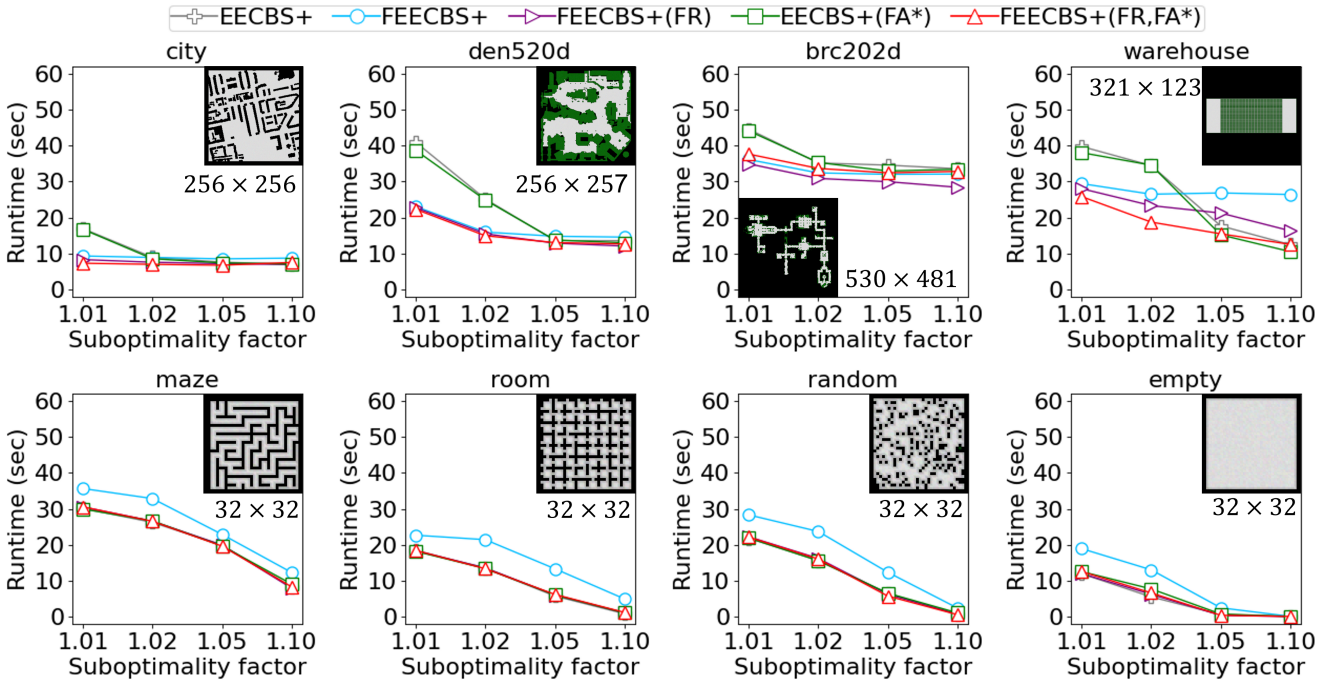


Figure 4: Average runtime on large (top row) and small (bottom row) maps. We include 60 seconds in the average for each unsolved MAPF instance.

low-level focal-A* search to avoid limitations of using flex distribution. Our empirical evaluation of FEECBS, the resulting version of EECBS, showed that flex distribution can improve the success rate of EECBS for large numbers of agents on large maps. Future work may include developing sophisticated distribution policies for flex.

Acknowledgements

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 research at Monash University was supported by the Australian Research Council under Discovery Grants DP190100013 and DP200100025 as well as a gift from Amazon.

References

Boyarski, E.; Bodic, P. L.; Harabor, D.; Stuckey, P. J.; and Felner, A. 2020a. F-Cardinal Conflicts in Conflict-Based Search. In *Proceedings of the International Symposium on Combinatorial Search (SOCS)*, 123–124.

Boyarski, E.; Felner, A.; Harabor, D.; Stuckey, P. J.; Cohen, L.; Li, J.; and Koenig, S. 2020b. Iterative-Deepening Conflict-Based Search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 4084–4090.

Boyarski, E.; Felner, A.; Sharon, G.; and Stern, R. 2015a. Don't Split, Try To Work It Out: Bypassing Conflicts in Multi-Agent Pathfinding. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 47–51.

Boyarski, E.; Felner, A.; Stern, R.; Sharon, G.; Tolpin, D.; Betzalel, O.; and Shimony, S. E. 2015b. ICBS: Improved Conflict-Based Search Algorithm for Multi-Agent Pathfinding. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 740–746.

Chan, S.-H.; Li, J.; Gange, G.; Harabor, D.; Stuckey, P. J.; and Koenig, S. 2021. ECBS with Flex Distribution for Bounded-Suboptimal Multi-Agent Path Finding. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, 159–161.

Felner, A.; Li, J.; Boyarski, E.; Ma, H.; Cohen, L.; Kumar, T. K. S.; and Koenig, S. 2018. Adding Heuristics to Conflict-Based Search for Multi-Agent Path Finding. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 83–87.

Gange, G.; Harabor, D.; and Stuckey, P. J. 2019. Lazy CBS: Implicit Conflict-Based Search Using Lazy Clause Generation. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 155–162.

Ho, F.; Salta, A.; Geraldes, R.; Goncalves, A.; Cavazza, M.; and Prendinger, H. 2019. Multi-Agent Path Finding for UAV Traffic Management. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 131–139.

Li, J.; Felner, A.; Boyarski, E.; Ma, H.; and Koenig, S. 2019a. Improved Heuristics for Multi-Agent Path Finding with Conflict-Based Search. In *Proceedings of the Inter-*

Maps	k	EECBS+		FEECBS+		FEECBS+(FR)		EECBS+(FA*)		FEECBS+(FR,FA*)	
		1.01	1.05	1.01	1.05	1.01	1.05	1.01	1.05	1.01	1.05
city	200	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**
	400	0.90	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**
	600	0.86	**0.94**	0.92	0.92	0.92	**0.94**	0.80	**0.94**	**0.94**	**0.94**
	800	0.44	**0.88**	0.84	0.84	0.84	0.86	0.46	0.86	**0.88**	**0.88**
den520d	200	0.92	**0.96**	**0.94**	0.94	0.92	**0.96**	**0.94**	**0.96**	**0.94**	**0.96**
	400	0.50	**0.92**	0.86	0.88	0.88	**0.92**	0.56	**0.92**	**0.92**	**0.92**
	600	0.00	0.86	0.74	0.80	**0.78**	0.86	0.10	**0.88**	**0.78**	**0.88**
	800	0.00	**0.86**	**0.26**	0.68	**0.26**	0.80	0.00	0.84	**0.26**	0.84
brc202d	200	0.98	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	400	0.44	**0.96**	**0.96**	**0.96**	**0.96**	**0.96**	0.40	**0.96**	**0.96**	**0.96**
	600	0.00	0.58	0.30	0.86	**0.42**	**0.92**	0.00	0.48	0.34	0.82
	800	0.00	0.00	0.00	0.02	0.00	**0.30**	0.00	0.28	0.00	0.04
warehouse	200	0.96	**1.00**	0.98	0.96	**1.00**	0.98	0.96	**1.00**	**1.00**	**1.00**
	400	0.66	0.94	0.82	0.84	0.86	0.90	0.68	**0.96**	**0.94**	**0.96**
	600	0.02	0.76	0.38	0.44	0.40	0.64	0.02	0.84	**0.52**	**0.92**
	800	0.00	0.60	0.00	0.14	**0.02**	0.28	0.00	**0.70**	0.00	0.68
maze	10	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	20	**0.94**	**1.00**	0.92	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**
	30	0.56	0.86	0.22	0.78	0.58	0.84	0.54	0.86	**0.60**	**0.88**
	40	**0.14**	0.50	0.02	0.32	0.12	0.50	**0.14**	0.52	0.12	**0.52**
	50	0.00	0.12	0.00	0.04	0.00	**0.16**	0.00	0.14	0.00	**0.16**
room	10	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	20	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	30	**0.94**	**1.00**	0.90	0.98	0.92	**1.00**	**0.94**	**1.00**	0.92	**1.00**
	40	**0.68**	**0.96**	0.36	0.82	0.66	0.92	0.66	0.94	0.64	0.92
	50	**0.08**	**0.76**	0.02	0.20	**0.08**	0.72	**0.08**	0.70	**0.08**	0.70
random	20	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	40	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	60	**0.94**	**1.00**	0.78	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**
	80	**0.42**	0.96	0.02	0.82	0.36	0.96	0.40	**0.98**	0.36	0.96
	100	0.00	0.64	0.00	0.22	0.00	0.64	0.00	0.62	0.00	**0.68**
empty	50	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	100	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	150	0.94	**1.00**	0.80	**1.00**	**0.98**	**1.00**	0.94	**1.00**	0.96	**1.00**
	200	**0.34**	**1.00**	0.02	0.88	**0.34**	**1.00**	0.32	**1.00**	0.30	**1.00**

Table 2: Success rates of for k agents on large (top 4) and small (bottom 4) maps with suboptimality factors 1.01 and 1.05. The suboptimality factors are shown in the row beneath the algorithms. Bold numbers indicate the maximum success rates given the map, the number of agents, and the suboptimality bound.

national Joint Conference on Artificial Intelligence (IJCAI), 442–449.

Li, J.; Gange, G.; Harabor, D.; Stuckey, P. J.; Ma, H.; and Koenig, S. 2020a. New Techniques for Pairwise Symmetry Breaking in Multi-Agent Path Finding. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 193–201.

Li, J.; Harabor, D.; Stuckey, P. J.; Ma, H.; and Koenig, S. 2019b. Disjoint Splitting for Multi-Agent Path Finding with Conflict-Based Search. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 279–283.

Li, J.; Harabor, D.; Stuckey, P. J.; Ma, H.; and Koenig, S. 2019c. Symmetry-Breaking Constraints for Grid-Based Multi-Agent Path Finding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 6087–6095.

Li, J.; Ruml, W.; and Koenig, S. 2021. EECBS: Bounded-Suboptimal Search for Multi-Agent Path Finding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 12353–12362.

Li, J.; Sun, K.; Ma, H.; Felner, A.; Kumar, T. K. S.; and Koenig, S. 2020b. Moving Agents in Formation in Congested Environments. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 726–734.

Li, J.; Zhang, H.; Gong, M.; Liang, Z.; Liu, W.; Tong, Z.; Yi, L.; Morris, R.; Pasareanu, C.; and Koenig, S. 2019d. Scheduling and Airport Taxiway Path Planning under Uncertainty. In *Proceedings of the AIAA Aviation Forum*, 1–7.

Ma, H.; Li, J.; Kumar, T. K. S.; and Koenig, S. 2017. Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 837–845.

Pearl, J.; and Kim, J. H. 1982. Studies in Semi-Admissible Heuristics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 392–399.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-Based Search for Optimal Multi-Agent Pathfinding. *Artificial Intelligence*, 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 *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, 151–159.

Thayer, J. T.; and Ruml, W. 2011. Bounded Suboptimal Search: A Direct Approach Using Inadmissible Estimates. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 674–679.