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) is the problem of finding collision-free paths for multiple agents that minimize the sum of path costs. 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 LB on C^* to find paths whose sum of path costs is at most $w \cdot LB$ in order to solve MAPF bounded-suboptimally. However, the costs of many paths are often much smaller than w times their minimum path costs, meaning that the sum of path costs is much smaller than $w \cdot C^*$. In this paper, we therefore propose Flexible EECBS (FEECBS), which uses a flex(ible) distribution of the path costs (that relaxes the requirement to find bounded-suboptimal paths 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. We address the drawbacks of flex distribution via techniques such as restrictions on the flex distribution, restarts of the high-level search with EECBS, and low-level focal-A* search. Our empirical evaluation shows that FEECBS substantially improves the efficiency of EECBS on MAPF instances with large maps and large numbers of agents.

Introduction

Multi-Agent Path Finding (MAPF) is the problem of finding collision-free paths for multiple agents, the set of which is called a *solution*. A solution of MAPF is optimal iff the sum of path costs (that is, travel times) is minimum, and bounded-suboptimal iff the sum of path costs is a user-specified suboptimality factor $w \geq 1$ 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 resolves collisions on the high level. Once CBS finds a collision between the paths of two agents on the high level, it generates two branches, where each branch forces one of the two agents 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 of MAPF is NP-hard (Yu and LaValle 2013) 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 solves MAPF bounded-suboptimally by replacing A* search with Explicit Estimation Search (EES) (Thayer and Ruml 2011) on the high level of CBS and A* search with focal search (Pearl and Kim 1982) on the low level of CBS. Focal search maintains a lower bound on the minimum path cost of each agent. It finds a bounded-suboptimal path for an agent by finding a path whose path cost is at most w times the lower bound on the minimum path cost of that agent. EES maintains a lower bound LB on the sum of path costs of an optimal solution, which it calculates using the lower bounds on the minimum path costs for all agents maintained by the low-level focal search. It finds a bounded-suboptimal solution by finding a solution whose sum of path costs is at most w times LB.

We call the difference between w times the sum of the lower bounds on the minimum path costs for all agents and the sum of the path costs of the current (perhaps not collision-free) paths for all agents the *flex*. The flex is thus the largest amount by which the sum of path costs of the current paths for all agents can increase when resolving collisions to guarantee that a bounded-optimal solution is found. EECBS distributes the flex in an inflexible way by requiring every current path of an agent to be bounded-suboptimal. In this paper, we propose Flexible EECBS (FEECBS) to distribute the flex among agents in a flexible way by allowing some current paths of agents not to be bounded-suboptimal as long as the set of paths is bounded-suboptimal. This allows FEECBS to resolve collisions and thus find bounded-suboptimal solutions faster than EECBS.

To ensure that the solution is bounded-suboptimal, EECBS and FEECBS need to increase LB until the sum of path costs of the solution is at most w times LB. When the flex assigned to an agent is large, the low-level focal search of FEECBS often finds a path for the agent with a sufficiently small path cost and then does not need to increase the lower bound on the minimum path cost of that agent. Thus, FEECBS often increases LB more slowly than EECBS. To address this issue, we propose restrictions on the flex distribution and rules for restarting the high-level search with EECBS. We also propose focal-A* search, where FEECBS switches to A* search on the low level after its focal search has generated more than a given number of nodes. This allows the low level of FEECBS to both increase the lower bound on the minimum path cost faster and return a path faster than EECBS (although that path can have more collisions with the paths of other agents). Our empirical evaluation shows that, although FEECBS has runtimes similar to 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 its speed-up techniques.

Multi-Agent Path Finding (MAPF)

We use the MAPF definition by Stern et al. (2019). A MAPF instance consists of an undirected graph and a set of k agents $\{a_1, \ldots, a_k\}$. 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. A *path* of an agent, starting at its start vertex and ending at its goal vertex, is a sequence of vertices indicating where the agent is at each timestep. Each agent permanently waits at its goal vertex after it completes its path. 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. The paths of two agents a_i and a_j have a vertex collision or, equivalently, *vertex conflict* iff the agents stay at the same vertex v at the same timestep t, denoted as $\langle a_i, a_j, v, t\rangle$. The paths of two agents a_i and a_j have an edge collision or, equivalently, *edge conflict* iff the agents traverse the same edge (u, v) in opposite directions at the same timestep t, denoted as $\langle a_i, a_j, u, v, t\rangle$. A *solution* is a set of conflict-free paths, one for each agent. We assume that a solution exists. 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 (Li, Ruml, and Koenig 2021) is a two-level algorithm that solves MAPF bounded-suboptimally for a user-specified suboptimality factor $w \geq 1$. 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. In addition, for each agent a_i, CT node N also contains a path that satisfies $constraints(N)$, whose cost is denoted as $c_i(N)$, and a lower bound $lb_i(N)$ on the cost $c_i^*(N)$ of a minimum-cost path for agent a_i that satisfies $constraints(N)$. 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 $c(N)$ denote $\sum_{i=1}^{k} c_i(N)$. 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 and 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 underestimates the difference between the minimum SoC of all solutions in the subtree of CT node N and $g(N)$. OPEN$_H$ is another regular open list of A* search and 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 of all CT nodes in CLEANUP$_H$ and $N_{\hat{f}}$ be a CT node in OPEN$_H$ with the minimum $\hat{f}$-value of all CT nodes in OPEN$_H$. For each CT node N in CLEANUP$_H$, $f(N)$ is a lower bound on the SoC of the best solution in its subtree. Since the CT node with an optimal solution is in one of the subtrees, there is a CT node N' in CLEANUP$_H$ with $f(N') \leq C^*$. Since $f(N_f)$ is the minimum f-value of all CT nodes in CLEANUP$_H$, it holds that $f(N_f) \leq f(N') \leq C^*$. Thus, $LB = f(N_f)$ is a lower bound on the SoC C^* of the optimal solution. 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 of all CT nodes in FOCAL$_H$. At each iteration, EECBS uses the following rules to select a CT node for expansion:

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

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

(E3) Else, expand N_f.

Thus, each expanded CT node N satisfies $c(N) \leq w \cdot LB$ and thus

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

The smaller the number of conflicts among the paths of some CT node N is, the more likely EECBS is to find a CT node with a solution in the subtree of CT node N. The smaller the $\hat{f}$-value of some CT node N is, the more likely it lies on a branch to a CT node with an optimal solution (Li, Ruml, and Koenig 2021) and thus the more likely EECBS is to find a CT node with an optimal solution in the subtree of CT node N. Thus, EECBS uses Rules (E1) and (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 to make the conditions of Rules (E1) and (E2) more likely to hold in the future and thus facilitate the expansion of CT nodes with those rules.

On the low level, given a CT node N, EECBS uses vertex-timestep nodes or, for short, v-t nodes $n = (v, t)$ 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 that 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 the v-t nodes in increasing order of an admissible cost function $f_i(n) = g_i(n) + h_i(n)$, where $g_i(n) = t$ 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 underestimates the number of timesteps for agent a_i to move from vertex v to its goal vertex g_i. FOCAL$_L$ contains those v-t nodes in OPEN$_L$ with $f_i(n) \leq \tau_i(N) = w \cdot f_{min,i}(N)$, where $f_{min,i}(N)$ is the minimum f-value of all v-t nodes n in OPEN$_L$ and $\tau_i(N)$ is called the *low-level focal threshold*, and sorts these v-t nodes $n = (v, t)$ in increasing order of the number of conflicts $d(n)$ with the paths of the other agents in CT node N when agent a_i moves from start vertex s_i at timestep 0 to vertex v at timestep t. At each iteration, focal search first updates FOCAL$_L$ and $f_{min,i}$, if necessary, and then expands the v-t node with the minimum d-value of all v-t nodes in FOCAL$_L$. It terminates when it expands a v-t node whose vertex is g_i.

For each v-t node n in OPEN$_L$, $f_i(n)$ is a lower bound on the cost of the best path in its subtree. Since the v-t node with the minimum-cost path is in one of the subtrees, there is a v-t node n' in OPEN$_L$ with $f_i(n') \leq c_i^*(N)$. Since $f_{min,i}(N)$ is the minimum f-value of all v-t nodes in OPEN$_L$, it holds that $f_{min,i}(N) \leq f_i(n') \leq c_i^*(N)$. Thus, $f_{min,i}(N)$ is a low-level lower bound on the cost $c_i^*(N)$ of the minimum-cost path for agent a_i that satisfies $constraints(N)$. Since the low-level search expands only v-t nodes from FOCAL$_L$, that is, v-t nodes $n = (v, t)$ with $f_i(n) \leq \tau_i(N)$ during the search, which includes those v-t nodes with $v = g_i$, $\tau_i(N)$ guarantees that the found path is bounded-suboptimal. Since the $f_i(n)$ of any v-t node n in FOCAL$_L$ is at most $\tau_i(N)$, EECBS always finds a bounded-suboptimal path whose cost $c_i(N)$ satisfies

$$f_{min,i}(N) \leq c_i(N) \leq \tau_i(N) = w \cdot f_{min,i}(N). \quad (2)$$

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,

$$lb_i(N) \leq c_i(N) \leq w \cdot lb_i(N) \leq w \cdot c_i^*(N). \quad (3)$$

Thus, the cost of CT node N satisfies

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

Table 1 provides a glossary of terms, notations, and definitions used in the paper.

Li, Ruml, and Koenig (2021) modified the speed-up techniques of CBS so that they can be used to speed up EECBS, namely prioritizing conflicts, bypassing conflicts, symmetry reasoning, and using the WDG heuristic as the admissible h-value of CT nodes. We provide more details of bypassing conflicts in the following.

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) $c(N) \leq w \cdot LB$, and (C4) $h_c(N) < h_c(\hat{N})$. EECBS uses (C1) because selecting CT nodes from CLEANUP$_H$ has a chance to increase LB and bypassing their conflicts would not increase LB. 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 $constraints(N)$ and $constraints(N)$ contains more constraints than $constraints(\hat{N})$, EECBS does not replace $lb_i(\hat{N})$ with $lb_i(N)$ when 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 $c_i(N) \leq w \cdot lb_i(N)$ for every agent a_i in CT node N. 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. Such a relaxation can be used for both resolving and bypassing conflicts.

Resolving Conflicts with Flex Distribution

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[1]

$$\Delta(N) = w \cdot g(N) - c(N). \quad (5)$$

[1]We use $g(N)$ instead of $f(N)$ to define the flex in Equation (5) because, otherwise, we cannot compute the flex over the other $k - 1$ agents in Equation (6). The reason for this issue is that $h(N) = f(N) - g(N)$ estimates the difference between the SoC of an optimal solution in the subtree of CT node N and $g(N)$ and does not specify the difference between the minimum cost of the path for each agent a_i and $c_i(N)$.

Term	Notation	Definition
The user-specified suboptimality factor	w	A solution is bounded-suboptimal iff the sum of costs of the paths of the solution is at most a factor of w away from minimum. A path is bounded-suboptimal iff its cost is at most a factor of w away from minimum.
The g-value of v-t node $n = (v, t)$ for agent a_i	$g_i(n)$	The number of timesteps for agent a_i to move from its start vertex s_i to vertex v.
The admissible h-value of v-t node $n = (v, t)$ for agent a_i	$h_i(n)$	An underestimate of the number of timesteps for agent a_i to move from vertex v to its goal vertex g_i.
The admissible f-value of v-t node n for agent a_i	$f_i(n)$	$g_i(n) + h_i(n)$.
Minimum low-level f-value of agent a_i in CT node N	$f_{min,i}(N)$	The minimum $f_i(n)$ of all v-t nodes n in OPEN$_L$ during the low-level search for agent a_i in CT node N.
The low-level lower bound on the path cost of agent a_i in CT node N	$lb_i(N)$	For EECBS, the value $f_{min,i}(N)$ after the low-level search for agent a_i in CT node N. For FEECBS, the value $\max\{f_{min,i}(N), lb_i(\hat{N})\}$ after the low-level search for agent a_i in CT node N, where $\hat{N}$ is the parent CT node of N.
The flex over all agents other than agent a_i	$\Delta_i(N)$	$\sum_{j=1,j\neq i}^{k} (w \cdot lb_j(N) - c_j(N))$.
The low-level focal threshold of agent a_t in CT node N	$\tau_i(N)$	FOCAL$_L$ consists of all v-t nodes n in CLEANUP$_L$/OPEN$_L$ with $\hat{f}(n) \leq \tau_i(N)$ during the low-level search for agent a_i in CT node N. For EECBS, $\tau_i(N) = wf_{min,i}(N)$. For FEECBS, $\tau_i(N) = wf_{min,i}(N) + \Delta_i(N)$.
The path cost of agent a_i in CT node N	$c_i(N)$	The number of timesteps for agent a_i to move from its start vertex s_i to its goal vertex g_i in CT node N.
The minimum path cost of agent a_i in CT node N	$c_i^*(N)$	The minimum number of timesteps for agent a_i to move from its start vertex s_i to its goal vertex g_i in CT node N.
The g-value of CT node N	$g(N)$	$\sum_{i=1}^{k} lb_i(N)$.
The admissible h-value of CT node N	$h(N)$	An underestimate of the difference between the minimum cost of all CT nodes with solutions in the subtree of CT node N and $g(N)$.
The inadmissible $\hat{h}$-value of CT node N	$\hat{h}(N)$	An estimate of the difference between the minimum cost of all CT nodes with solutions in the subtree of CT node N and $g(N)$.
The admissible f-value of CT node N	$f(N)$	$g(N) + h(N)$.
The inadmissible $\hat{f}$-value of CT node N	$\hat{f}(N)$	$g(N) + \hat{h}(N)$.
The cost of CT node N, that is, the sum of costs of the paths of CT node N	$c(N)$	$\sum_{i=1}^{k} c_i(N)$.
The sum of costs of the paths of an optimal solution	C^*	
The flex of CT node N	$\Delta(N)$	$w \cdot g(N) - c(N) = \sum_{j=1}^{k} (w \cdot lb_j(N) - c_j(N))$.
The top CT node in CLEANUP$_H$	N_f	The CT node with the minimum f-value in CLEANUP$_H$/OPEN$_H$.
The lower bound on the sum of costs of the paths of an optimal solution	LB	$f(N_f)$.
The top CT node in OPEN$_H$	$N_{\hat{f}}$	The CT node with the minimum $\hat{f}$-value in CLEANUP$_H$/OPEN$_H$.
The top CT node in FOCAL$_H$	N_{h_c}	The CT node with the minimum h_c-value in FOCAL$_H$.
The limit on the number of CT nodes selected for expansion by Rule (E3)	T_N	If FEECBS selects more than T_N CT nodes for expansion with Rule (E3), then we restart the high-level search with EECBS.
The number of generated v-t nodes during the low-level search for agent a_i in CT node N	$\eta_i(N)$	
The limit on the number of generated v-t nodes during the low-level search for agent a_i in CT node N	$T_i(N)$	If a low-level focal search of FEECBS generates more than $T_i(N)$ v-t nodes, then we switch to an A* search for the remainder of the low-level search. In this paper, $T_i(N) = \kappa \cdot \eta_i(\tilde{N})$, where $\kappa \geq 1$ is a user-specified constant and $\tilde{N}$ is the closest ancestor CT node of CT node N where FEECBS found a path for agent a_i.

Table 1: The glossary with terms, notations, and definitions.

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

$$\begin{aligned}\Delta_i(N) &= w \cdot \sum_{j=1,j\neq i}^{k} lb_j(N) - \sum_{j=1,j\neq i}^{k} c_j(N) \\ &= w \cdot (g(N) - lb_i(N)) - (c(N) - c_i(N)).\end{aligned} \tag{6}$$

According to Equations (5) and (6), we know that

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

We need to ensure that Inequality (4) holds for all CT nodes N. Thus, we need to ensure that

$$\Delta(N) \overset{(5)}{\geq} 0, \tag{8}$$

which, in turn, indicates that we need to ensure that $\Delta_i(N) \geq -w \cdot lb_i(N) + c_i(N)$ (see Equation (7)). Therefore, after replanning the path of agent a_i in CT node N, we need to ensure that

$$c_i(N) \leq w \cdot lb_i(N) + \Delta_i(N). \tag{9}$$

EECBS uses $\tau_i(N) = w \cdot f_{min,i}(N)$ during the low-level search and sets $lb_i(N)$ to $f_{min,i}(N)$ after the low-level search finishes. By using flex during the low-level search, FEECBS can use a relaxed $\tau_i(N) = w \cdot f_{min,i}(N) + \Delta_i(N)$ and continue to set $lb_i(N)$ to $f_{min,i}(N)$ after the low-level search finishes to ensure that Inequality (9) holds. Furthermore, since $constraints(\hat{N}) \subsetneq constraints(N)$ ensures that $c_i^*(\hat{N}) \leq c_i^*(N)$, $lb_i(\hat{N})$ is, like $lb_i(N)$, a lower bound on $c_i^*(N)$. We can redefine $\tau_i(N)$ and $lb_i(N)$ in the following way to ensure that $lb_i(N)$ is non-decreasing along any branch in the CT, which results in larger low-level lower bounds on the path costs on the low level and thus also larger g-values on the high level, which speeds up the searches on both levels. We let FEECBS use

$$\tau_i(N) = w \cdot \max\{f_{min,i}(N), lb_i(\hat{N})\} + \Delta_i(N) \tag{10}$$

and set $lb_i(N)$ to $\max\{f_{min,i}(N), lb_i(\hat{N})\}$ after the low-level search finishes. More importantly, this new definition guarantees FOCAL$_L$ to be non-empty during the low-level search, while the old one does not. More specifically, the paths of the other agents do not change during the low-level search for agent a_i, so

$$\begin{aligned} \Delta_i(N) &= \Delta_i(\hat{N}) \\ &\overset{(7)}{=} \Delta(\hat{N}) - w \cdot lb_i(\hat{N}) + c_i(\hat{N}) \\ &\overset{(8)}{\geq} -w \cdot lb_i(\hat{N}) + lb_i(\hat{N}), \end{aligned} \tag{11}$$

where we use $c_i(\hat{N}) \geq lb_i(\hat{N})$ in the derivation of the last inequality since $lb_i(\hat{N})$ is the low-level lower bound on the path cost $c_i(\hat{N})$. Therefore,

$$\begin{aligned} \tau_i(N) &= w \cdot \max\{f_{min,i}(N), lb_i(\hat{N})\} + \Delta_i(N) \\ &\overset{(11)}{\geq} w \cdot \max\{f_{min,i}(N), lb_i(\hat{N})\} - w \cdot lb_i(\hat{N}) + lb_i(\hat{N}) \\ &= w \cdot \max\{f_{min,i}(N) - lb_i(\hat{N}), 0\} + lb_i(\hat{N}) \\ &\geq \max\{f_{min,i}(N) - lb_i(\hat{N}), 0\} + lb_i(\hat{N}) \\ &\geq f_{min,i}(N). \end{aligned} \tag{12}$$

Thus, $\tau_i(N) \geq f_{min,i}(N)$, which indicates that the first node in OPEN$_L$ during the low-level search is always in FOCAL$_L$, that is, FOCAL$_L$ is always non-empty and the low-level search is thus guaranteed to find a path for agent a_i if one exists.

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.

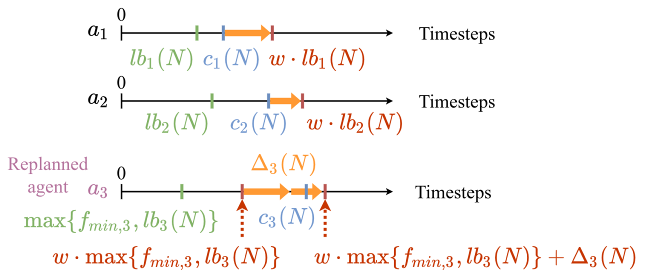

Figure 1: An illustrative example of how positive flex increases the low-level focal threshold. Suppose FEECBS generates CT node N and replans the path of agent a_3 with positive flex over agents a_1 and a_2. The green bars are the low-level lower bounds, the blue bars are the path costs, and the red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

w	1.01	1.02	1.05	1.10
EECBS	66.7	54.2	30.4	13.0
FEECBS	43.0	27.0	10.9	8.96

Table 2: The sums of the LB improvements of EECBS and FEECBS over 7,000 MAPF instances on 8 different maps for suboptimality factors $w = \{1.01, 1.02, 1.05, 1.10\}$, which show that EECBS has higher LB improvements than FEECBS.

	Rule (E1)	Rule (E2)	Rule (E3)	Sum
EECBS	25	5	16	46
FEECBS	11	10	52	63

Table 3: The numbers of CT nodes selected by Rules (E1), (E2), and (E3) for expansion in Figure 2.

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 $c_i(N)$ of each new path is still no larger than $w \cdot lb_i(\hat{N})$. For FEECBS, since we relax the low-level focal threshold of the path of each agent a_i by using flex distribution, the path cost does not need to satisfy this inequality 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

$$c(N) \leq w \cdot g(\hat{N}). \tag{13}$$

We compare FEECBS with and without bypassing conflicts over 7,000 MAPF instances on 8 different maps using the setup described in the Empirical Evaluation section. We add all other speed-up techniques to both algorithms, namely prioritizing conflicts, symmetry reasoning, and using the WDG heuristic. The result shows that FEECBS solves 50 additional MAPF instances when using bypassing conflicts.

Limitations of Flex Distribution

When finding a path for agent a_i in a CT node N whose parent CT node is $\hat{N}$, FEECBS reduces the number of con-

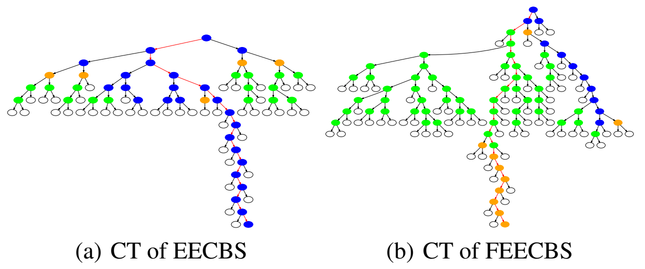
(a) CT of EECBS (b) CT of FEECBS

Figure 2: The CTs of EECBS and FEECBS for the same MAPF instance. The blue, orange, and green 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 (yet) been expanded. The arrows are the directed edges connecting CT nodes with their child CT nodes, where the red arrows indicate a path that finds a solution.

flicts by relaxing the low-level focal threshold. However, the relaxed low-level focal threshold has two limitations.

FEECBS sometimes raises LB more slowly than EECBS. Suppose that $w \cdot lb_i(\hat{N}) < c_i^*(N) \le w \cdot lb_i(\hat{N}) + \Delta_i(N)$. For EECBS, we have

$$w \cdot lb_i(\hat{N}) < c_i^*(N) \le c_i(N) \overset{(3)}{\le} w \cdot lb_i(N), \qquad (14)$$

which results in $lb_i(\hat{N}) < lb_i(N)$, meaning that the low-level lower bound on the path cost of agent a_i increases from CT node $\hat{N}$ to N. For FEECBS, it is possible to find a path with cost $c_i(N) \le w \cdot lb_i(\hat{N}) + \Delta_i(N)$, in which case $lb_i(\hat{N})$ may be equal to $lb_i(N)$ and the low-level lower bound on the path cost of agent a_i remains the same from CT node $\hat{N}$ to N. Thus, FEECBS results in a smaller LB improvement than EECBS, as shown in Table 2, where the LB improvement is defined as the difference in the beginning and at the end of the high-level search. Since FEECBS uses flex, $c(N)$ may be larger for FEECBS than that for EECBS. The combination of the poor LB improvement and the large $c(N)$ may result in fewer CT nodes that satisfy $c(N) \le w \cdot LB$ for FEECBS than for EECBS. Thus, the first limitation (L1) is that FEECBS tends to select CT nodes by Rule (E3) for expansion more frequently than EECBS on the high level since $c(N_{h_c})$ and $c(N_{\hat{f}})$ may be larger than $w \cdot LB$ due to the lower LB improvement. For instance, Figure 2 and Table 3 show the CTs and the numbers of CT nodes selected for expansion by EECBS and FEECBS with $w = 1.05$ and all speed-up techniques while solving the same MAPF instance (the "room" map in the Empirical Evaluation section with 40 agents), where FEECBS selects more CT nodes by Rule (E3) for expansion.

On the low level, if all 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 likely expands more v-t nodes n that satisfy $d(n) < m$. As shown in Figure 3,

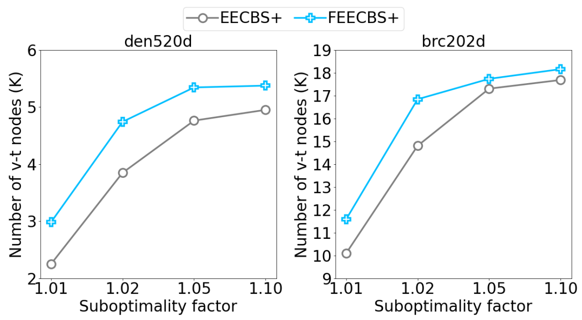

Figure 3: Average number of v-t nodes in FOCAL$_L$ per low-level search of EECBS+ (EECBS with all speed-up techniques) and FEECBS+ (FEECBS with all speed-up techniques) over 200 MAPF instances on two maps.

FEECBS+ (FEECBS with all speed-up techniques) has indeed a higher average number of v-t nodes in FOCAL$_L$ per low-level search than EECBS+ (EECBS with all speed-up techniques) over 200 MAPF instances on two different maps (namely, the "den520d" and "brc202d" maps in the Empirical Evaluation section) with the number of agents ranging from 200 to 800 in increments of 200.

To alleviate the limitations, we add flex restrictions to FEECBS such that it does not always distribute flex among the agents and restarts the high-level search with EECBS if the number of CT nodes selected for expansion by Rule (E3) becomes large. Also, we use low-level focal-A* search, that switches the low-level search from a focal search to an A* search if the number of expanded v-t nodes becomes large. We provide details on these approaches in the next two sections.

Flex Restrictions and Restart with EECBS

When finding a path for agent a_i in CT node N whose parent CT node is $\hat{N}$, if $\Delta_i(\hat{N}) < 0$, then we have to use flex distribution in CT node N in order to satisfy Inequality (4). Otherwise, we set up restrictions on using flex distribution to make it more likely that $lb_i(N) > lb_i(\hat{N})$. In particular, we prohibit FEECBS from distributing the flex and ask it to use the original focal search instead 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*, that is, all combinations of the minimum-cost paths of the two conflicting agents lead to this conflict.[2] We use Restriction (R1) to avoid that the low-level search distributes the entire flex to one agent in the beginning of the high-level search. We use Restrictions (R2) and (R3) since both selecting CT nodes by Rule (E3) for expansion and resolving cardinal conflicts make it more likely that $lb_i(N) > lb_i(\hat{N})$.

Although we set up restrictions on using flex distribution, EECBS can still increase LB faster than FEECBS. Thus, when using FEECBS, we will restart the search with EECBS

[2]We reuse the technique for prioritizing conflicts described in (Li, Ruml, and Koenig 2021) to identify cardinal conflicts.

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 $g_i$
1  Compute $\Delta_i(N)$ and $T_i(N)$
2  root v-t node $r \leftarrow (s_i, 0)$
3  $\eta_i(N) \leftarrow 1$
4  $f_{min,i}(N) \leftarrow f(r)$
5  if use flex restrictions and $\Delta_i(N) \geq 0$ then $\delta \leftarrow 0$
6  else $\delta \leftarrow \Delta_i(N)$
7  $\tau_i(N) \leftarrow w \cdot \max\{f_{min,i}(N), lb_i(\hat{N})\} + \delta$
8  OPEN$_L \leftarrow \phi$, FOCAL$_L \leftarrow \phi$
9  InsertNode($r$)
10 while OPEN$_L$ is not empty do
11     if $\eta_i(N) \leq T_i(N)$ then
12         $n \leftarrow$ Pop the top v-t node from FOCAL$_L$
13         Delete $n$ from OPEN$_L$
14     else $n \leftarrow$ Pop the top v-t node from OPEN$_L$
15     if IsGoal($n$) then return ExtractPath($n$)
16     $neighbors \leftarrow$ ExpandNode($n$)
17     for $n' \in neighbors$ do
18         if $n'$ satisfies $constraints(N)$ then
19             $\eta_i(N) \leftarrow \eta_i(N) + 1$
20             InsertNode($n$)
21     if $f_{min,i}(N) < f$(top v-t node in OPEN$_L$) then
22         $f_{min,i}(N) \leftarrow f$(top v-t node in OPEN$_L$)
23         if $\eta_i(N) \leq T_i(N)$ then
24             $\tau'_i \leftarrow w \cdot \max\{f_{min,i}(N), lb_i(\hat{N})\} + \delta$
25             UpdateFocal($\tau_i(N), \tau'_i$)
26             $\tau_i(N) \leftarrow \tau'_i$
27 return "No path satisfies $constraints(N)$."
```

if the number of CT nodes that are continuously selected by Rule (E3) for expansion exceeds a user-specified CT node limit $T_N \geq 0$.

Low-Level Focal-A* Search

To alleviate the limitations of FEECBS, we switch from focal search to A* search during the low-level search for a path of agent a_i in CT node N when the number of generated v-t nodes exceeds the v-t node limit $T_i(N)$. That is, we then select the v-t node n with the minimum $f_i(n)$ in OPEN$_L$ for expansion and ignore FOCAL$_L$, which tends to raise $f_{min,i}(N)$ faster than focal search and thus alleviates Limitation (L1). Since we ignore FOCAL$_L$, it also alleviates Limitation (L2). We can also use low-level focal-A* search for EECBS since switching to A* search does not depend on flex.

However, determining a good value for $T_i(N)$ is instance-dependent and thus difficult. Here, we provide an intuitive guideline that works empirically. Suppose that FEECBS found a path for 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 when planning the path for agent a_i in CT node $\tilde{N}$ as $\eta_i(\tilde{N})$. The number of generated v-t nodes for CT node $\tilde{N}$ can be used to estimate the number of generated v-t nodes for CT node N. We thus define the v-t node limit $T_i(N)$ as

$$T_i(N) = \kappa \cdot \eta_i(\tilde{N}), \tag{15}$$

where $\kappa \geq 1$ is a user-specified constant. That is, the v-t node limit increases as the number of generated v-t nodes in the ancestor CT node increases.

Algorithm 1 shows the pseudo-code of the low-level focal-A* search with flex distribution and flex restrictions. We first compute $\Delta_i(N)$ and $T_i(N)$ and initialize the root v-t node $(s_i, 0)$, the number of generated v-t nodes $\eta_i(N)$, and $f_{min,i}(N)$ [Lines 1 to 4]. If the flex restrictions are triggered and the flex over all other agents is non-negative, then we ignore $\Delta_i(N)$ [Lines 5 to 7]. If $\eta_i(N) \leq T_i(N)$, then we use focal search [Lines 11 to 13] and update FOCAL$_L$ [Line 25]. Otherwise, we switch to A* by expanding v-t nodes from OPEN$_L$ [Line 14] and ignoring FOCAL$_L$. Function InsertNode(n) inserts v-t node n into OPEN$_L$. If $f_i(n) \leq \tau_i(N)$, then we insert it into FOCAL$_L$ as well. Function IsGoal(n) checks if vertex v of the v-t node $n = (v, t)$ is the goal vertex g_i. Function ExtractPath(n) generates a path by starting at v-t node n and repeatedly moving from a v-t node to its parent until the root v-t node is reached. Function ExpandNode(n) generates the set of v-t nodes (v', t') where v' is either vertex v or a neighboring vertex of v and $t' = t + 1$. To terminate the low-level search within a finite runtime limit, we avoid any wait action if all other agents have reached their goal vertices before timestep t. Function UpdateFocal($\tau_i(N), \tau'_i$) inserts v-t nodes n with $\tau_i(N) < f_i(n) \leq \tau'_i(N)$ from OPEN$_L$ into FOCAL$_L$.

Empirical Evaluation

We evaluate the algorithms on eight 4-neighbor 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 warehouse grid map (`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 of 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 100 to 500 in increments of 100. 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 1.10. The algorithms are implemented in C++, and the experiments are conducted with CentOS Linux on an AMD EPYC 7302 16-Core Processor with a memory limit of 16 GB. Our comparison metrics are the success rate, which is the percentage of MAPF

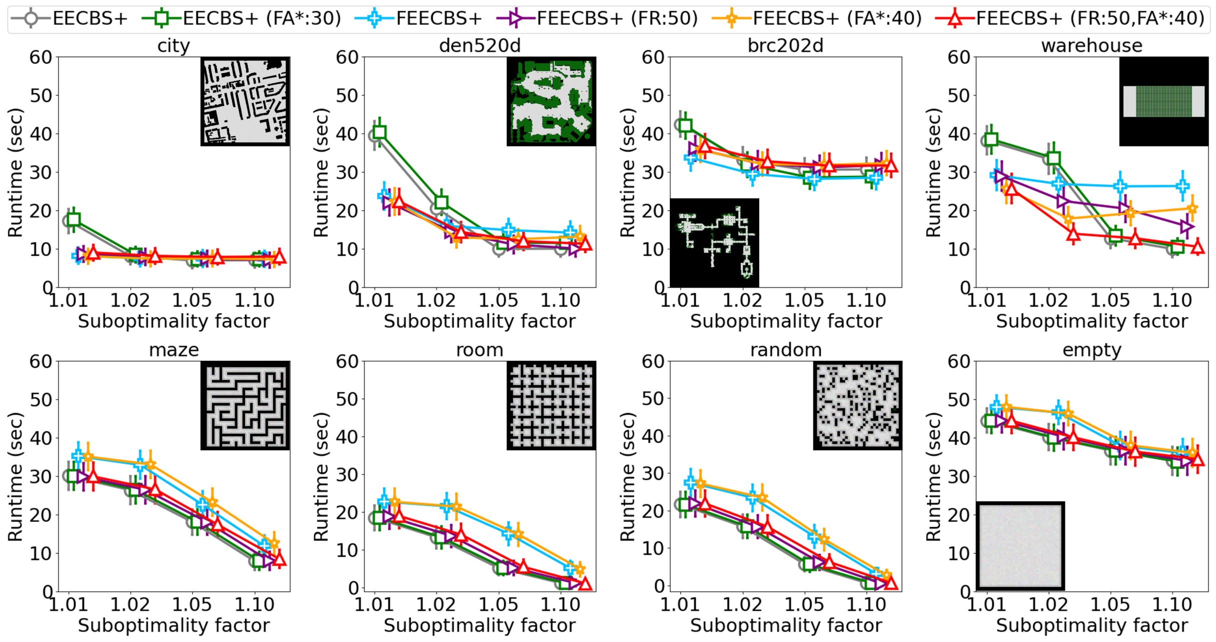
Figure 4: Average runtimes on large (top row) and small (bottom row) maps. We include 60 seconds in the average for each MAPF instance where the runtime limit was reached. The data points in the figures show the average runtimes over all MAPF instances for the corresponding maps as described in the Empirical Evaluation section, with the bars being the confidence intervals. We shift the points horizontally to avoid overlaps.

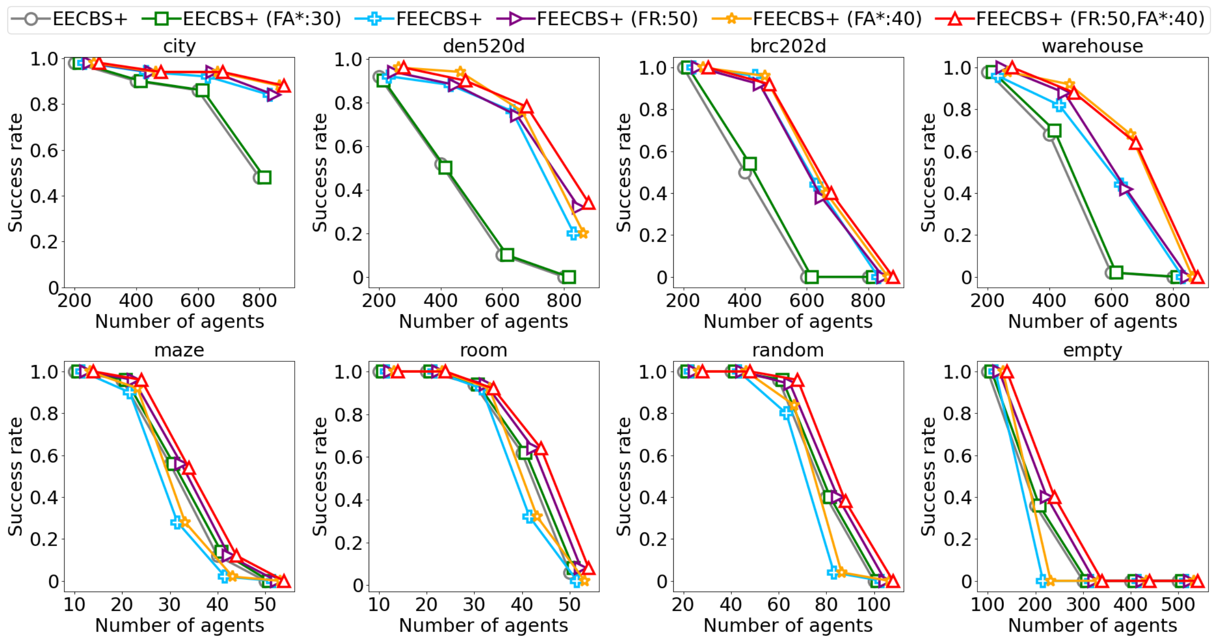
Figure 5: Success rates on large (top row) and small (bottom row) maps for suboptimality factor $w = 1.01$. The data points are the success rates over 50 MAPF instances (both "even" and "random" scenarios) for a given number of agents in the map. We shift the points horizontally to avoid overlaps.

instances solved within the runtime limit of 60 seconds per MAPF instance, and the runtime.

We evaluate EECBS+ (EECBS with all speed-up techniques), FEECBS+ (FEECBS with all speed-up techniques), FEECBS+ (FR:50) (FEECBS with all speed-up techniques, flex restrictions, and restart with EECBS with $T_N = 50$, which results in the lowest average runtime over all MAPF instances among $T_N = \{10, 50, 100\}$), EECBS+ (FA*:20)

(EECBS with all speed-up techniques and low-level focal-A* search with $\kappa = 20$, which results in the lowest average runtime over all MAPF instances among $\kappa = \{10, 20, 30, 40\}$), and FEECBS+ (FR:50,FA*:30) (FEECBS with all speed-up techniques, flex restrictions, restart with EECBS with $T_N = 50$, and low-level focal-A* search with $\kappa = 30$, which results in the lowest average runtime over all MAPF instances among $\kappa = \{10, 20, 30, 40\}$ with $T_N = 50$). Figure 4 shows the average runtimes for different suboptimality factors. On large maps, if the suboptimality factor is small (that is, $w = 1.01$), 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, more flex can be distributed, and thus FEECBS+ improves less over EECBS+ due to Limitations (L1) and (L2). However, with flex restrictions and restart with EECBS, the success rates of FEECBS+ (FR:50) are competitive with the ones of EECBS+ on small maps, and, with low-level focal-A* search, both EECBS+ (FA*:20) and FEECBS+ (FR:50,FA*:30) show improvements in runtime. Figure 5 shows the success rates for suboptimality factor $w = 1.01$, where FEECBS+ (FR:50, FA*:30) outperforms EECBS+ and EECBS+ (FA*:20) especially on the warehouse map.

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 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 grants 1409987, 1724392, 1817189, 1837779, 1935712, and 2112533 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 Multi-Agent 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 International 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 Multi-Agent 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 Path Finding: 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.

Yu, J.; and LaValle, S. M. 2013. Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 1443–1449.