# ECBS with 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, USA
[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. CBS is a leading optimal two-level MAPF solver whose low level plans optimal paths for single agents and whose high level runs a best-first search on a Constraint Tree (CT) to resolve the collisions between the paths. ECBS, a bounded-suboptimal variant of CBS, speeds up CBS by reducing the number of collisions that need to be resolved on the high level. It achieves this by generating bounded-suboptimal paths with fewer collisions with the paths of the other agents on the low level and expanding bounded-suboptimal CT nodes that contain fewer collisions on the high level. In this paper, we propose Flexible ECBS (FECBS) that further reduces the number of collisions that need to be resolved on the high level by using looser suboptimal bounds on the low level while still providing bounded-suboptimal solutions. Instead of requiring the cost of each path to be bounded-suboptimal, FECBS requires only the overall cost of the paths to be bounded-suboptimal, which gives us the freedom to distribute the cost leeway among different agents according to their needs. Our empirical results show that FECBS can solve more MAPF instances than state-of-the-art ECBS variants within 5 minutes.

## Multi-Agent Path Finding (MAPF)

MAPF is the problem of finding collision-free paths on a graph for $k$ agents $\{a_1, \ldots, a_k\}$, each with a start vertex and a goal vertex. At every discretized timestep, an agent can either *move* to an adjacent vertex or *wait* at its current vertex. A path for agent $a_i$ is a sequence of vertices indicating where agent $a_i$ is at each timestep, with its *path cost* being the number of timesteps needed by agent $a_i$ to reach its goal vertex and stay there. A *collision* occurs when two agents occupy the same vertex or traverse the same edge at the same timestep. A *solution* is a set of collision-free paths, one for each agent. An *optimal solution* is a solution with the minimum *sum of costs (SoC)* of the paths.

## Enhanced Conflict-Based Search (ECBS)

Enhanced Conflict-Based Search (ECBS) (Barer et al. 2014) is a two-level MAPF solver that is guaranteed to find a *bounded-suboptimal* solution, i.e., a solution whose SoC is at most $w \cdot C^*$, where $w$ is a user-specified suboptimality factor and $C^*$ is the SoC of an optimal solution.

On the high level, ECBS searches a *Constraint Tree* (CT). A CT node $N$ contains a set of constraints (for resolving collisions), a lower bound $lb_i(N)$ on the cost $c_i^*(N)$ of the optimal path of each agent $a_i$ that satisfies the constraints, and a path with cost $c_i(N) \leq w \cdot lb_i(N)$ for each agent $a_i$ that satisfies the constraints. The root CT node contains no constraints. Given a collision between a pair paths in a CT node chosen for expansion, ECBS resolves it by generating two child CT nodes, each with an additional constraint that prohibits one of the colliding agents from using the contested vertex or edge at the colliding timestep. ECBS maintains an open list $\text{OPEN}_\text{H}$ (like A*) and a focal list $\text{FOCAL}_\text{H}$ that contains all CT nodes $N$ in $\text{OPEN}_\text{H}$ that satisfy

$$\sum_{i=1}^{k} c_i(N) \leq w \cdot LB, \tag{1}$$

where $LB = \min_{N \in \text{OPEN}_\text{H}} \{\sum_{i=1}^{k} lb_i(N)\}$ is guaranteed to be a lower bound on $C^*$. In each iteration, ECBS first updates $\text{FOCAL}_\text{H}$ and $LB$ if necessary, then selects the CT node in $\text{FOCAL}_\text{H}$ with the smallest number of collisions, and resolves one collision by expansion. ECBS terminates when the selected CT node has no collisions. Since the SoC of the paths of all CT nodes in $\text{FOCAL}_\text{H}$ are at most $w \cdot LB$, ECBS always finds a solution with a SoC of at most $w \cdot C^*$.

When generating a CT node $N$ with an additional constraint on agent $a_i$, ECBS first sets the constraints, the lower bounds, and the paths of all agents other than agent $a_i$ in $N$ to those in its parent CT node and then finds the path and the lower bound of agent $a_i$ in $N$ via a search on its low level.

On the low level, ECBS searches in the vertex-timestep space to find both a bounded-suboptimal path for an agent that satisfies the constraints and a lower bound on the cost of its optimal path that satisfies the constraints. ECBS maintains an open list $\text{OPEN}_\text{L}$ (like A*) and a focal list $\text{FOCAL}_\text{L}$ that contains all nodes $n$ in $\text{OPEN}_\text{L}$ whose $f$-values are at most the threshold $w \cdot f_{\min}$, where $f_{\min} = \min_{n \in \text{OPEN}_\text{L}} \{f(n)\}$ is guaranteed to be a lower bound on $c_i^*(N)$. In each iteration, ECBS first updates $\text{FOCAL}_\text{L}$ and $f_{\min}$ if necessary and then expands the node in $\text{FOCAL}_\text{L}$ with the smallest number of collisions with the paths of the other agents in CT node $N$. Since the $f$-values of all nodes in

FOCAL$_L$ are at most $w \cdot f_{\min}$, ECBS always finds a bounded-suboptimal path with cost $c_i(N) \le w \cdot lb_i(N)$, where $lb_i(N)$ is set to $f_{\min}$ when the low-level search terminates, that is, $lb_i(N)$ is a lower bound on $c_i^*(N)$.

Thus, for every agent $a_i$ in every CT node $N$, we have $c_i(N) \le w \cdot lb_i(N)$, which implies that

$$w \cdot \sum_{i=1}^{k} lb_i(N) - \sum_{i=1}^{k} c_i(N) \ge 0. \tag{2}$$

So, ECBS can always select a CT node from FOCAL$_H$ in each iteration on the high level because FOCAL$_H$ contains at least the CT node $N_{LB} = \arg\min_{N \in \text{OPEN}_H} \{\sum_{i=1}^{k} lb_i(N)\}$ as Inequality (2) ensures that $N_{LB}$ satisfies Inequality (1).

## Flexible ECBS (FECBS)

On the low level, ECBS always finds a bounded-suboptimal path for an agent that satisfies the constraints. However, since the bounded suboptimality of ECBS is guaranteed by only expanding CT nodes that satisfy Inequality (1) on the high level, we can relax the bounded suboptimality on the low level to further reduce the number of collisions that need to be resolved on the high level. We thus propose Flexible ECBS (FECBS) that, rather than guaranteeing the cost of each path to be bounded-suboptimal, only guarantees the SoC of the paths in each CT node to be bounded-suboptimal, that is, Inequality (2) to hold (otherwise, FOCAL$_H$ might be empty). We refer to the left-hand side of Inequality (2) as the *flex* over the $k$ agents. Intuitively, when FECBS replans a path for agent $a_i$ in a CT node $N$, if the flex $\Delta_i(N) = w \cdot \sum_{i' \neq i} lb_{i'}(N) - \sum_{i' \neq i} c_{i'}(N)$ over the other $k-1$ agents is positive, then it can satisfy Inequality (2) by finding a path with a cost of at most $w \cdot lb_i(N) + \Delta_i(N)$ even if the cost is larger than the threshold $w \cdot lb_i(N)$ of ECBS.

Formally, FECBS differs from ECBS only in the threshold of FOCAL$_L$. That is, when replanning the path of agent $a_i$ in a CT node $N$ whose parent CT node is $\hat{N}$, FOCAL$_L$ contains all nodes in OPEN$_L$ whose $f$-values are at most $w \cdot \max\{f_{\min}, lb_i(\hat{N})\} + \Delta_i(N)$ (instead of $w \cdot f_{\min}$). We use $\max\{f_{\min}, lb_i(\hat{N})\}$ instead of $f_{\min}$ here because, otherwise, the new threshold might be smaller than $f_{\min}$, which can result in an empty low-level FOCAL. More specifically, if FECBS found paths for some agents with costs larger than $w$ times their lower bounds at ancestor CT nodes of $N$, then $\Delta_i(N)$ might be negative and $w \cdot f_{\min} + \Delta_i(N)$ might be smaller than $f_{\min}$. But, because of Inequality (2), we know that $\Delta_i(N) = w \cdot \sum_{i' \neq i} lb_{i'}(N) - \sum_{i' \neq i} c_{i'}(N) = w \cdot \sum_{i' \neq i} lb_{i'}(\hat{N}) - \sum_{i' \neq i} c_{i'}(\hat{N})) = (w \cdot \sum_{i=1}^{k} lb_i(\hat{N}) - \sum_{i=1}^{k} c_i(\hat{N})) - w \cdot lb_i(\hat{N}) + c_i(\hat{N}) \ge -w \cdot lb_i(\hat{N}) + lb_i(\hat{N})$. Thus, our new threshold satisfies $w \cdot \max\{f_{\min}, lb_i(\hat{N})\} + \Delta_i(N) \ge w \cdot \max\{f_{\min}, lb_i(\hat{N})\} - w \cdot lb_i(\hat{N}) + lb_i(\hat{N}) = w \cdot \max\{f_{\min} - lb_i(\hat{N}), 0\} + lb_i(\hat{N}) \ge \max\{f_{\min} - lb_i(\hat{N}), 0\} + lb_i(\hat{N}) = \max\{f_{\min}, lb_i(\hat{N})\} \ge f_{\min}$, which ensures that FOCAL$_L$ is never empty.

FECBS usually uses a larger threshold of FOCAL$_L$ than ECBS, so, to avoid it finding paths that involve unnecessary waits or detours, we add a tie-breaking rule to FOCAL$_L$: If multiple nodes in FOCAL$_L$ have the same number of collisions, we prefer one with the smallest $f$-value. When the low level of FECBS terminates, FECBS assigns the found path to agent $a_i$ in $N$ and sets $lb_i(N)$ to $\max\{f_{\min}, lb_i(\hat{N})\}$. $lb_i(N)$ is a lower bound on $c_i^*(N)$ because $lb_i(\hat{N})$ is a lower bound on $c_i^*(\hat{N})$, which is at most $c_i^*(N)$ since the additional constraint on agent $a_i$ cannot make the cost of an optimal path of agent $a_i$ smaller. Therefore, $c_i(N) \le w \cdot lb_i(N) + \Delta_i(N)$, which implies that Inequality (2) holds, that is, FOCAL$_H$ is never empty. Since FECBS plans paths for agents one at a time at the root CT node and does not know the flex over the other agents in advance, FECBS uses the new threshold only when replanning paths at non-root CT nodes.

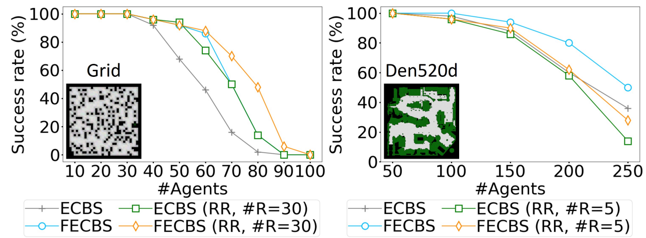


Figure 1: Success rates of MAPF solvers on different maps.

## Empirical Evaluation

We use two 4-neighbor grids, namely a $32 \times 32$ grid map with 20% blocked cells (`Grid`) and a $257 \times 256$ game map (`Den520d`), and both the “even” and “random” scenarios from the MAPF benchmark suite (Stern et al. 2019). We let the suboptimality factor be 1.05 and 1.01 for the `Grid` and `Den520d` maps, respectively. We use $\max\{f_{\min}, lb_i(\hat{N})\}$ as the threshold of FOCAL$_L$ and the $f$-values to break ties in our ECBS implementation (which sped up ECBS).

ECBS with the Rapid Randomized Restart (RR) technique (Cohen et al. 2018) is a state-of-the-art ECBS variant. Given a user-specified number of runs #R and a time limit $T$ (in seconds), the RR technique restarts the search every $T$/#R seconds. We denote ECBS and FECBS with the RR technique as ECBS (RR) and FECBS (RR), respectively. We define *best #R* as the value of #R in the set $\{5, 20, 30, 40\}$ that leads to the highest *success rate*, i.e, the percentage of MAPF instances solved within 5 minutes. Figure 1 shows the success rates versus the number of agents. We only show ECBS (RR) and FECBS (RR) with their best #R. FECBS (RR) dominates on the `Grid` map, and FECBS dominates on the `Den520d` map.


## 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

Barer, M.; Sharon, G.; Stern, R.; and Felner, A. 2014. Sub-optimal Variants of the Conflict-Based Search Algorithm for the Multi-Agent Pathfinding Problem. In *Proceedings of the 7th Annual Symposium on Combinatorial Search (SoCS)*, 19–27.

Cohen, L.; Koenig, S.; Kumar, T. K. S.; Wagner, G.; Choset, H.; Chan, D.; and Sturtevant, N. R. 2018. 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)*, 1909–1911.

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 12th International Symposium on Combinatorial Search (SoCS)*, 151–159.