A Conceptual Agile Conflict-Based Search Framework for Scalable Multi-Agent Pathfinding in Dense Environments

Shafakhatullah Khan Mohammed *shafakhat91@gmail.com*
Department of Computer Science Engineering,
Maulana Azad University, Jodhpur,
Rajasthan, 342802, India.

Pallavi Singhal *pratap.pallavi@gmail.com*
Department of Computer Science Engineering,
Maulana Azad University, Jodhpur,
Rajasthan, 342802, India.

Abstract

This paper presents theoretically proved Agile CBS algorithm, an extension of the traditional Conflict-Based Search which is customized to support bounded-suboptimal solutions for Multi-Agent Pathfinding problems. The method uses flexible assignment of intermediate targets and adaptive resolution of constraints, focusing to improve scalability, especially in environments with high agent density. Unlike conventional CBS, which resolves conflicts reactively, Agile CBS decomposes long-term objectives into incremental sub-goals, enabling progressive planning while maintaining bounded suboptimality guarantees. The algorithm utilizes a constraint tree for high-level search and A* for low-level path computation, thereby reducing planning overhead in scenarios where optimal solvers encounter computational challenges. While theoretical in nature, this framework provides a foundation for developing scalable classical AI-based MAPF solvers.

Keywords: Conflict-Based Search, Heuristic Search, Multi-Agent Pathfinding, Bounded Suboptimality, Collision Avoidance.

1. INTRODUCTION

In robotics, logistics, and traffic systems, where multiple agents must navigate shared spaces without colliding while maximizing goals like travel time or distance, Multi-Agent Pathfinding (MAPF) tackles coordination issues (Stern et al., 2021). Complex conflict resolution techniques that strike a balance between computational effectiveness and solution quality are necessary for the best MAPF solutions.

The primary distinction between single-agent and multi-agent pathfinding is how each shapes the problem formulation and the solution approaches. Single-Agent Pathfinding (SAPF) determines optimal routes between graph nodes using algorithms like A* (Hart et al., 1968), where the evaluation function $f(n) = g(n) + h(n)$ guides search efficiently. When the heuristic h is admissible, A* guarantees optimal solutions (Dechter & Pearl, 1985).

MAPF extends this model to multiple agents, each agent has its own start and goal locations. The objective is computing collision-free paths that minimize aggregate cost, typically the sum of individual path costs. This problem is NP-hard (Yu & LaValle, 2013), with state space complexity growing exponentially with agent count. Direct application of A* to the joint state space yields branching factors of b^k for k agents, making pure search-based approaches computationally prohibitive for large-scale instances.

1.1 Collision Types in MAPF

MAPF solutions must address multiple collision types:

- *Vertex collisions:* Multiple agents occupying the same location simultaneously
- *Edge collisions:* Agents traversing the same edge in opposite directions
- *Following collisions:* An agent entering a recently vacated cell
- *Wait collisions:* Conflicts arising from stationary agents

These collision patterns necessitate specialized detection and resolution strategies (Bing Ai et al., 2021).

1.2 Research Methodology

The methodology adopted in this paper is deductive and conceptual. It involves the theoretical proposal of the Agile CBS framework by extending the known architecture of Conflict-Based Search (CBS) with novel components—dynamic sub-goal decomposition and bounded-suboptimal guarantees. The paper proceeds by formalizing the framework's definitions (Section 3.2), outlining its two-level architecture (Section 3.3), and theoretically deriving its complexity analysis (Section 3.4), without relying on empirical data collection or analysis in this initial phase.

2. RELATED WORK

MAPF algorithms are categorized into optimal, bounded-suboptimal, and suboptimal solvers based on solution quality guarantees and computational requirements.

2.1 Optimal Algorithms

Conflict-Based Search (CBS) (Sharon et al., 2014) employs a two-level architecture: the high-level maintains a Constraint Tree (CT) where nodes represent constraint sets, while the low-level computes individual agent paths using A* under current constraints. When conflicts arise (vertex conflict $\pi_i(t) = \pi_j(t)$ or edge conflict), CBS branches the CT by adding constraints that prevent agents from occupying conflicting locations. This process continues until a conflict-free solution emerges. CBS achieves optimality regarding sum-of-costs but exhibits exponential time complexity $O(2^c \cdot n \log n)$ where c represents conflict count and n is agent count.

Increasing Cost Tree Search (ICTS) (Sharon et al., 2012) operates over cost vectors $C = \langle c_1, c_2, \ldots, c_k \rangle$ where each c_i represents agent $a_i's$ individual path cost. ICTS explores solution space incrementally from minimal sum-of-costs, expanding nodes through single-agent cost increments while checking path combinations for conflicts. This enables effective pruning but incurs exponential complexity in agent count.

A* with Independence Detection (A* + ID) (Ryan, 2008) initially plans paths independently, then progressively merges conflicting agents into meta-agents that plan over joint state spaces $O(b^m)$ where m is merged agent count. This achieves $O(k \cdot n \log n)$ complexity in low-conflict scenarios but degrades exponentially as conflicts increase.

Modified A* (M*) (Wagner & Choset, 2011) employs locality-aware planning, expanding joint configuration space only for conflicting agents using dynamic collision sets. With worst-case complexity $O(|V|^{|C|})$ where $|C|$ is collision set size, M* performs efficiently in sparse-conflict environments but struggles with dense agent populations.

2.2 Bounded-Suboptimal Algorithms

Bounded-suboptimal algorithms guarantee solutions within a factor w of optimal cost ($Cost \leq w \cdot OPT$), trading optimality for computational efficiency.

Enhanced CBS (ECBS)(Barer et al., 2021) extends CBS using focal search at the high level, prioritizing nodes within factor w of minimum cost in OPEN. Low-level search employs suboptimal A* with inflated heuristics $f = g + w \cdot h$. ECBS maintains $O(2^c \cdot n \log n)$ complexity while ensuring w-suboptimality. Unlike ECBS, which optimizes node selection in the existing constraint tree, Agile CBS introduces a mechanical change via dynamic sub-goal decomposition, aiming to

structurally reduce the number of conflicts (c) itself, thus addressing a fundamental scalability bottleneck of CBS in dense environments.

Explicit Estimation CBS (EECBS)(Li et al., 2021) incorporates heuristic estimates for both individual agents and joint interactions, explicitly estimating total solution cost including potential conflicts:

$$f(n) = \Sigma\, cost(a_i) + h\_conflict(n)$$

This improved cost estimation changes the order of node expansion. It cuts down on the computational load while keeping the bounded-suboptimality guarantees intact. While EECBS improves search effectiveness by better estimating the total cost, Agile CBS focuses on reducing the length of the paths being considered at any one time through incremental sub-goals, leading to localized and potentially fewer conflicts to resolve, a distinct approach to managing conflict density in dense settings.

Anytime Algorithms, including **Anytime Weighted A* (AWA*)** (Stern et al., 2014) and **Anytime Repairing A* (ARA*)** (Chan et al., 2023), quickly give initial suboptimal solutions and then refine them over time to get closer to optimality. AWA* uses a fixed inflation weight w, while ARA* gradually lowers ε from high starting values. It reuses previous search information using repair methods. Agile CBS's decomposition of long-term objectives into incremental sub-goals enables a form of progressive refinement, conceptually aligning with the anytime behavior of these algorithms. The use of parameter w is also directly derived from bounded-suboptimal and anytime search principles.

2.3 Suboptimal Algorithms

Priority-Based Search (PBS) (Barer et al., 2021) plans sequentially, assigning total priority ordering over agents, with lower-priority agents being constrained by higher-priority paths. In sparse settings, this achieves $O(k \cdot n \log n)$ complexity, but depending on priority ordering, it may result in deadlocks or less-than-ideal solutions.

By using fixed planning windows with reservation tables for local coordination, Windowed Hierarchical Cooperative A* (WHCA*) (Silver, 2021) significantly reduces computation while sacrificing completeness outside of planning horizons.

Using local push and swap operations, Push and Swap (De Wilde et al., 2013, Luna & Bekris, 2011, Sajid et al., 2021) sequentially moves agents to goals in $O(n^2)$ time, but frequently produces extremely suboptimal paths because of a lack of global coordination.

With velocity-based decentralized planning and O(1) complexity per agent per frame, Optimal Reciprocal Collision Avoidance (ORCA) (Van Den Berg et al., 2011) allows for real-time navigation but may result in deadlocks in confined spaces without global planning.

Recent innovations include HiMAP (Tang et al., 2024) that incorporates learned heuristics, MAPF-LNS2 (Huang et al., 2022) that uses machine learning-guided large neighborhood search, and LaCAM (Okumura, 2023) for quick pathfinding. Dynamic environment adaptability is addressed by real-time replanning techniques (Zhang et al., 2024).

3. AGILE CBS: CONCEPTUAL FRAMEWORK

3.1 Motivation

Existing CBS variants face scalability challenges in dense environments due to extensive conflict resolution. Agile CBS addresses this through dynamic sub-goal decomposition, enabling localized planning that reduces conflict density while maintaining bounded-suboptimality guarantees.

3.2 Formal Definition

Let $G = (V, E)$ be a graph and $A = \{a_1, a_2, \ldots, a_k\}$ be k agents, where each agent $a_i \in A$ is defined by tuple $\langle s_i, g_i \rangle$ with $s_i, g_i \in V$ denoting start and goal vertices. Each agent seeks path $\pi_i: \mathbb{N} \to V$ such that $\pi_i(0) = s_i$, $\exists\, T_i \in \mathbb{N}$ where $\pi_i(T_i) = g_i, and\, \pi_i(t+1) \in \{\pi_i(t)\} \cup neighbors(\pi_i(t))$.

Agile CBS operates over a Constraint Tree (CT) τ where each node N contains:

- Constraint set C_N (vertex/edge constraints per agent)
- Sub-goal set $S_N = \{g_{i1}, g_{i2}, \ldots, g_{ik}\}$
- Solution $\pi_N = \langle \pi_1, \pi_2, \ldots, \pi_k \rangle$ consistent with C_N
- Cost cost(N) = Σ len(π_i)
- Heuristic $h(N)$ estimating remaining conflict resolution cost

3.3 Two-Level Architecture

High-Level Search: Best-first search over CT using evaluation function $f(N) = cost(N) + w \cdot h(N)$ where $w \geq 1$ is the suboptimality bound. Node expansion proceeds by conflict detection and constraint branching.

Low-Level Search: For each agenta_i, A* computes shortest path from current position to current sub-goal g_i under constraints C_N. Upon conflict detection in π_N, the high-level creates child nodes with disjoint constraints for conflicting agents, potentially reassigning local sub-goals to avoid repetitive conflict patterns.

3.4 Complexity Analysis

Agile CBS introduces sub-goal planning phases, trading single large searches for multiple smaller searches over fewer agents. Let m be sub-goals per agent and g be goal advancement phases.

Time Complexity: $O(g \cdot b^{k'})$ where $k' < k$ is agents per sub-goal phase, resulting from spatial partitioning that reduces conflict scope compared to standard CBS complexity $O(2^c \cdot n \log n)$.

Space Complexity: $O(g \cdot k \cdot |V| \cdot T)$ where paths may be discarded between sub-goal phases for memory efficiency.

3.5 Algorithm

Algorithm 1: Agile Conflict-Based Search (ACBS)

Input:

- Graph $G = (V, E)$,
- Agents $A = \{a_1, a_2, \ldots, a_k\}$ with start vertices $s_i \in V$ and goal vertices $g_i \in V$
- Bound $w \geq 1$.

Output:

- Conflict-free paths $\pi_N = \langle \pi_1, \pi_2, \ldots, \pi_k \rangle$, one path per agent.

Algorithm:

1. Initialize the root Conflict Tree (CT) node:
   - Constraints $C_N = \emptyset$
   - Sub-goals $S_N = \{g_{i1}, g_{i2}, \ldots, g_{ik}\}$.
2. For each agent a_i, assign an initial sub-goal g_{i1} minimizing distance to final goal g_i:
$$g_{i1} = \arg\min_{v \in S_N} \text{dist}(s_i, v)$$
3. Compute initial paths for all agents using A* considering constraints C_N:

root.paths[i] ← A*(s_i, g_{i1}, C_N)

4. Set root cost:

$$\text{root.cost} \leftarrow \sum_{i=1}^{k} \text{len}(\pi_i)$$

5. Initialize OPEN priority queue ordered by:

$$f(N) = \text{cost}(N) + w \cdot h(N)$$

where $h(N)$ is a heuristic estimate of the remaining cost.

6. Insert the root node into $OPEN$.
7. While $OPEN \neq \emptyset$:
   - Pop node $P \leftarrow pop(OPEN)$.
   - If $P.paths$ is conflict-free:
     - If all agents reached their final goals g_i, return $P.paths$.
     - Else, update sub-goals S_N with next sub-goals for agents and:P.paths[i] ← A*$(\text{current\_pos}_i, \text{new } g_i, C_N)$

$$P.\text{cost} \leftarrow \sum_{i=1}^{k} \text{len}(\pi_i)$$

     - Insert P into $OPEN$.
   - Else, detect a conflict:(a_i, a_j, v, t)indicating agents a_i and a_j collide at vertex v at time t.
     - For each agent in $\{a_i, a_j\}$:
       A. Create a child node new_node as a copy of P.
       B. Add constraint forbidding the agent from being at vertex$v$ at time t:

$$\text{new\_node.}C_N \leftarrow \text{new\_node.}C_N \cup \{(agent, v, t)\}$$

       C. Recompute the path respecting the new constraints:

$$\text{new\_node.paths}[agent] \leftarrow \text{A}^*(\text{current\_pos}_{agent}, g_{agent}, \text{new\_node.}C_N)$$

       D. If a valid path exists:
          - Update new_node.cost ← $\sum_{i=1}^{k} \text{len}(\pi_i)$
          - Insert new_node into $OPEN$.

4. DISCUSSION

Agile CBS presents a conceptual framework combining hierarchical planning with dynamic sub-goal assignment. By breaking down agent objectives into smaller targets, ACBS allows for localized conflict resolution. This reduces the computational load compared to handling conflicts on a global scale. The bounded-suboptimality guarantee through parameter w offers a solid theoretical foundation while providing practical flexibility.

The suggested architecture naturally supports parallel processes through independent sub-goal planning phases and step-by-step path refinement. This setup may improve scalability in warehouse robotics, coordination of autonomous vehicles, and air traffic management, where real-time performance is essential.

4.1 Theoretical Advantages

Agile CBS presents a theoretical solution to the scalability challenges of traditional CBS in dense environments through several key advantages:

Reduced Conflict Density: By breaking long-term goals into smaller, localized sub-goals, the planning horizon and the spatial scope of potential conflicts are significantly reduced. This is expected to lower the critical conflict count (c), addressing the exponential complexity $O(2^c \cdot n \log n)$ that hinders CBS in dense scenarios.

Progressive Refinement: The decomposition enables the generation of intermediate, non-final path segments, which supports an anytime behavior where initial, coarse solutions can be progressively refined.

Bounded Guarantees: The integration of the suboptimality bound w (used in the high-level search function $f(N) = cost(N) + w \cdot h(N)$ maintains a theoretical guarantee on solution quality relative to the optimal cost.

Adaptive Replanning: The framework's ability to reassign local sub-goals upon conflict detection allows it to adaptively target and resolve conflicts in congested regions.

4.2 Limitations

As a conceptual framework, ACBS requires empirical validation to confirm theoretical advantages. Key challenges include:

- Sub-goal selection strategies affecting performance
- Overhead from multiple planning phases
- Heuristic design for conflict estimation
- Scalability limits in extremely dense scenarios

5. FUTURE WORK

Empirical validation will test ACBS on standard MAPF benchmarks including Sturtevant's 4-connected grid maps (Sturtevant, 2012) with 50-100 agents in dense scenarios such as Dragon Age maps and urban layouts. Evaluation metrics include:

- Runtime performance
- Solution cost $\Sigma\, Cost(\pi_i)$
- Suboptimality ratio $Cost\_ACBS\ /\ Cost\_OPT$
- Scalability with agent count k

Comparative analysis against CBS, ECBS, LaCAM (Okumura, 2023), MAPF-LNS2 (Huang et al., 2022), HiMAP (Tang et al., 2024), and real-time replanning methods (Zhang et al., 2024) will assess efficiency in reducing conflict density and planning overhead. Implementation will formalize sub-goal assignment strategies and integrate with existing CBS variants to enable comprehensive performance evaluation.

6. CONCLUSION

The research question addressed by this paper is: "**How can the Conflict-Based Search framework be adapted to achieve scalable Multi-Agent Pathfinding solutions in dense environments while maintaining bounded-suboptimality guarantees?**"

Agile CBS introduces a bounded-suboptimal MAPF framework employing dynamic sub-goal decomposition and adaptive constraint resolution. By combining hierarchical planning with localized conflict handling, ACBS offers potential computational advantages in dense environments. The theoretical foundation balances solution quality through bounded suboptimality with efficiency through progressive refinement. While empirical validation remains necessary, this conceptual approach demonstrates promising directions for scalable classical AI-based MAPF solving. The framework's emphasis on sub-goal-driven, plan-merging strategies represents a viable path toward addressing real-world multi-agent coordination challenges requiring both optimality guarantees and computational efficiency.

7. REFERENCES

Bing Ai, Jiuchuan Jiang, Shoushui Yu, and Yichuan Jiang. (2021) 'Multi-Agent Path Finding with heterogeneous edges and roundtrips', *Knowledge-Based Systems, 234(5)*, 107554.

Barer, M., Sharon, G., Stern, R. & Felner, A. (2021) 'Suboptimal variants of the Conflict-Based search algorithm for the Multi-Agent pathfinding problem', In Proceedings of the International Symposium on Combinatorial Search (SoCS), 5(1), pp. 19-27.

Chan, S., Stern, R., Felner, A. & Koenig, S. (2023) 'Greedy Priority-Based search for suboptimal Multi-Agent path finding', In Proceedings of the International Symposium on Combinatorial Search (SoCS), 16(1), pp. 11-19.

Dechter, R. & Pearl, J. (1985) 'Generalized best-first search strategies and the optimality of A*', *Journal of the ACM, 32(3)*, pp. 505-536.

De Wilde, B., Ter Mors, A.W. & Witteveen, C. (2013) 'Push and rotate: cooperative multi-agent path planning', In Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 87-94.

Hart, P., Nilsson, N. & Raphael, B. (1968) 'A formal basis for the heuristic determination of minimum cost paths', *IEEE Transactions on Systems Science and Cybernetics, 4(2)*, pp. 100-107.

Huang, T., Li, J., Koenig, S. & Dilkina, B. (2022) 'Anytime Multi-Agent path finding via Machine Learning-Guided large neighborhood search', In Proceedings of the AAAI Conference on Artificial Intelligence, 36(9), pp. 9368-9376.

Li, J., Ruml, W. & Koenig, S. (2021) 'EECBS: A Bounded-Suboptimal Search for Multi-Agent Path Finding', In Proceedings of the AAAI Conference on Artificial Intelligence, 35(14), pp. 12353-12362.

Okumura, K. (2023) 'LACAM: Search-Based Algorithm for Quick Multi-Agent Pathfinding', In Proceedings of the AAAI Conference on Artificial Intelligence, 37(10), pp. 11655-11662.

Ryan, M.R. (2008) 'Exploiting subgraph structure in Multi-Robot Path Planning', *Journal of Artificial Intelligence Research, 31*, pp. 497-542.

Sharon, G., Stern, R., Goldenberg, M. & Felner, A. (2012) 'The increasing cost tree search for optimal multi-agent pathfinding', *Artificial Intelligence, 195*, pp. 470-495.

Sharon, G., Stern, R., Felner, A. & Sturtevant, N.R. (2014) 'Conflict-based search for optimal multi-agent pathfinding', *Artificial Intelligence, 219*, pp. 40-66.

Silver, D. (2021) 'Cooperative pathfinding', In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), 1(1), pp. 117-122.

Stern, R., Felner, A., Van Den Berg, J., Puzis, R., Shah, R. & Goldberg, K. (2014) 'Potential-based bounded-cost search and Anytime Non-Parametric A*', A*rtificial Intelligence, 214*, pp. 1-25.

Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., Li, J., Atzmon, D., Cohen, L., Kumar, T.K., Barták, R. & Boyarski, E. (2021) 'Multi-Agent Pathfinding: Definitions, variants, and benchmarks', Proceedings of the International Symposium on Combinatorial Search (SoCS), 10(1), pp. 151-158.

Sturtevant, N.R. (2012) 'Benchmarks for Grid-Based Pathfinding', *IEEE Transactions on Computational Intelligence and AI in Games, 4(2)*, pp. 144-148.

Tang, H., Berto, F., Ma, Z., Hua, C., Ahn, K. & Park, J. (2024) 'HiMAP: Learning Heuristics-Informed Policies for Large-Scale Multi-Agent Pathfinding', In Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 2498-2500.

Van Den Berg, J., Guy, S.J., Lin, M. & Manocha, D. (2011) 'Reciprocal N-Body collision avoidance', *Springer Tracts in Advanced Robotics*, pp. 3-19.

Wagner, G. & Choset, H. (2011) 'M*: A complete multirobot path planning algorithm with performance bounds', In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3260-3267.

Yu, J. & LaValle, S. (2013) 'Structure and intractability of optimal Multi-Robot Path planning on graphs', In Proceedings of the AAAI Conference on Artificial Intelligence, 27(1), pp. 1443-1449.

Zhang, Y., Chen, Z., Harabor, D., Bodic, P.L. & Stuckey, P.J. (2024) 'Planning and Execution in Multi-Agent Path Finding: Models and Algorithms', In Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 34, pp. 707-715.