# Relative Investigation of Multi-Agent Path Finding Solvers

**Sameer Shastri[1*], Dr.Nagenrea Tripathi[2], Dr. S. L. Sinha[3], Dr. S.P.Shukla[4], Dr. Supriya Tripathi[5]**

[1] Department of Electronics & Communication Engineering, Government Engineering College, Raipur, Chhattisgarh, India

[2] Department of Electrical Engineering, Bhilai Institute of Technology Durg, Chhattisgarh, India

[3] Department of Mechanical Engineering National Institute of Technology Raipur, Chhattisgarh, India

[4] Department of Electrical Engineering, Bhilai Institute of Technology Durg, Chhattisgarh, India

[5] Department of Electrical Engineering, Bhilai Institute of Technology Durg, Chhattisgarh, India

Email: [1] sameershastri@bitdurg.ac.in, [2] ntripathi@bitdurg.ac.in, [3] lsinha.mech@nitrr.ac.in, [4] sp.shukla@bitdurg.ac.in, [5] supriya.tripathi@bitdurg.ac.in

## Abstract

This paper investigates the use of iterative refinement in multi-agent pathfinding (MAPF), a path-planning problem for multiple robots. We analyze and compare several path planning algorithms, including ICBS, HCA, PIBT, CBS, and WHCA, to identify the most effective approach to finding optimal paths for multiple agents. Using these algorithms, we develop a suboptimal MAPF solver that quickly generates initial solutions, which we then refined through an iterative selection of a subset of agents and the application of an optimal MAPF solver. Our study provides valuable insights into the effectiveness of iterative refinement in MAPF and can help researchers identify the most efficient approach to this problem. These findings have practical applications in areas such as robotics, logistics, and transportation, where efficient path planning for multiple agents is essential for achieving optimal performance. This paper investigates the use of iterative refinement in multi-agent path-finding (MAPF), a path-planning problem for multiple robots. We analyse and compare several path-planning algorithms, including ICBS, HCA, PIBT, CBS, and WHCA, to identify the most effective approach to finding optimal paths for multiple agents. Using these algorithms, we develop a sub-optimal MAPF solver that quickly generates initial solutions, which we then refined through an iterative selection of a subset of agents and the application of an optimal MAPF solver. Our study provides valuable insights into the effectiveness of iterative refinement in MAPF and can help researchers identify the most efficient approach to this problem. The process of determining an optimal or near-optimal path between two points in a given environment while avoiding obstacles and other limitations is known as path planning. It is a significant issue in several areas, including robotics, autonomous cars, computer graphics, and video games. The goal of path planning is to find a feasible path that meets the following criteria: it connects the starting point to the goal point, it avoids obstacles

and other environmental constraints, it optimises a specific performance metric such as distance, time, or energy, and it takes into account the dynamic nature of the environment and any changes that may occur during execution. These findings have practical applications in agents that are essential for achieving optimal performance.
Keywords: MAPF (Multi-Agent Path Finding) CBS (Conflict-Based Search), ICBS (Improved Conflict-Based Search), PIBT (Parallel Iterative Bidirectional Decoupling), HCA (Hierarchical Cooperative A*), and WHCA (Windowed Hierarchical Cooperative A*).

## 1. Introduction

Path planning is the process of finding an optimal or near-optimal path between two points in a given environment while avoiding obstacles and other constraints. It is an important problem in various fields, including robotics, autonomous vehicles, computer graphics, and video games. The objective of path planning is to find a feasible path that satisfies the following criteria: it connects the starting point to the goal point, it avoids obstacles and other constraints in the environment, it optimizes a specific performance metric such as distance, time, or energy, and it takes into account the dynamic nature of the environment and any changes that may occur during execution. Path planning algorithms can be classified into different categories based on their approach and the type of problem they solve.

Path planning is a critical problem-solving technique that involves finding an optimal or near-optimal path between two points in a given environment while avoiding obstacles and other constraints. The potential applications of path planning are vast and include automated warehouse, intersection management, airport surface operation, automated parking, and video games. Path planning algorithms are used in these applications to provide safe and efficient navigation, generate realistic and natural-looking motion, and enable users to navigate through virtual environments. The development of effective path-planning algorithms has the potential to significantly improve the performance, efficiency, and safety of various autonomous systems and applications. The major problem in multi-agent path planning methods is the complexity of the problem as the number of agents increases. In multi-agent path planning, multiple agents need to navigate through the same environment simultaneously, and the paths of different agents may interfere with each other. This interference can lead to conflicts such as collisions or deadlock situations, which need to be resolved to find a feasible path for all agents. The complexity of the problem and the coordination and communication requirements make multi-agent path planning a challenging problem to solve, and research is ongoing to develop effective algorithms and techniques to address these challenges. Conversely, it is possible to generate solutions that are not optimal but can be obtained quickly (6 to 10). These solutions may not prioritize quality but are still preferable to having no solution at all due to waiting for optimal solvers that may not meet the deadline. Optimal solvers cannot guarantee to find solutions within a specified time frame. By obtaining workable solutions rapidly, we can utilize the remaining time until the deadlines to perform iterative improvement. This is the fundamental concept behind anytime algorithms which is to produce a feasible solution when interrupted, and whose quality will increase with time. In our research, we have investigated several algorithms to obtain the initial solution, such as Conflict-Based Search (CBS), Improved Conflict-Based Search (ICBS), Priority Inheritance with Backtracking (PIBT), Hierarchical Cooperative A*

(HCA*), and Windowed Hierarchical Cooperative A* (WHCA*). Our study involves a thorough comparison of these methods based on their ability to provide a collision-free path. We have evaluated these methods and recommended the most suitable approach to achieve an optimal solution. Although iterative refinement is a promising approach for Multi-Agent Path Finding (MAPF), it has been underutilized due to the challenge of incrementally improving a known solution. In the context of local search, this involves identifying a favorable neighborhood solution. Therefore, we introduce an anytime framework of iterative refinement for MAPF. Our research paper focuses on comparing various path-planning algorithms used for finding an initial path and optimizing it. Path planning algorithms are essential techniques used to find an optimal or near-optimal path between two points while avoiding obstacles and other constraints. In our study, we explore the effectiveness of different path-planning algorithms in finding an initial path, which is then optimized through iterative refinement. By comparing and evaluating the performance of these algorithms, we aim to determine the most efficient and effective method to achieve an optimal path for various applications such as robotics, autonomous vehicles, and computer graphics.

## 2. Literature Review

Our research paper proposes an anytime framework for multi-agent pathfinding (MAPF) that combines existing solvers. The framework first uses a sub-optimal solver to quickly obtain an initial feasible solution, followed by using an optimal solver to iteratively refine the solution by selecting a subset of agents and improving their paths while keeping others fixed. The refinement process solves a sub-problem that is much smaller than the original, making it faster. Our research aims to compare various path-planning algorithms used for discovering an initial path and enhancing it through iterative refinement. We investigate the efficiency of the different path-planning algorithms in detecting an initial path, which is subsequently optimized. By assessing and comparing the performance of these algorithms, we intend to determine the most efficient and effective technique to attain an optimal path for different applications, including autonomous vehicles, robotics, and computer graphics.

To achieve this aim, we examined various algorithms, including Conflict-Based Search (CBS), Improved Conflict-Based Search (ICBS), Parallel Iterative Bidirectional Decoupling (PIBT), Hierarchical Cooperative A* (HCA*), and Windowed Hierarchical Cooperative A* (WHCA*), to obtain the initial solution for collision-free pathfinding. We conducted a comprehensive comparison of these methods and evaluated their effectiveness. Our research recommends the most appropriate approach for achieving an optimal solution.

## 3. Methodology

To create an anytime version of the optimal MAPF algorithm, Standley and Korf. expanded on their earlier work. A form of A-based anytime algorithm was investigated by Cohen et al. and utilised in MAPF. An anytime MAPF solver, X assumes sparse circumstances, i.e., sparsely dispersed agents in fields with low collision probability. These three techniques iteratively tighten the constraints to eventually get optimal solutions after initially searching for non-optimal answers by loosening some constraints. They are each dependent on a particular solver, which has the disadvantage that they might not find initial solutions within a reasonable amount of time, returning nothing. Surynek researched the neighbourhood repair rules for a pebble motion on graphs; It is MAPF-adaptable. Improvements are made through ad hoc local adjustments, therefore the solution still contains redundant instances of apriori unidentified patterns.

## 4. Related Contributions :

### 4.1.Hierarchical Cooperative A* (HCA)

The use of heuristics based on abstractions of the state space is a common approach to improving pathfinding algorithms. There are two generic methods for improving such heuristics. The first involves precomputing all distances in the abstract space and storing them in a pattern database, but this is not feasible for dynamic maps such as those encountered in Cooperative Pathfinding. The second approach is to use Hierarchical A*, where abstract distances are computed on demand. This is more suitable for dynamic environments. In 2017, Yu et al. proposed an extension to the HCA* algorithm called Weighted Hierarchical Cooperative A* (WHCA*) to address the issue of suboptimal solutions. The authors introduced a weight factor to the heuristic function used in HCA* to improve the quality of the obtained solution. When testing HCA* and WHCA* In maze-like environments, research has shown that the success rate of all methods is nearly the same when the number of agents is small. However, as the number of agents increases, the success rate begins to decrease for some algorithms. WHCA* has been found to have the best success rate compared to other methods.

### 4.2.Windowed Hierarchical Cooperative A* - (WHCA)

WHCA* is an algorithm used for multi-agent path planning in robotics. It is designed to coordinate the movements of multiple agents in an environment to reach their respective destinations while avoiding collisions. The algorithm operates in a hierarchical manner, consisting of high-level and low-level planning stages. In the high-level stage, agents plan their paths within a specific time window. The time window represents a discrete period during which the agents' paths are planned. The high-level planner takes into account the current and predicted positions of other agents to determine a collision-free path for each agent within its assigned time window.

In the low-level stage, a local planner, typically using the A* algorithm, is employed to generate the detailed trajectory for each agent within its time window. The local planner considers the current state of the environment, including the positions of other agents, to avoid collisions while following the planned high-level path. WHCA* is known for its ability to handle complex scenarios with multiple agents and dynamic obstacles. It provides a centralized approach for coordination and collision avoidance, ensuring that agents can navigate safely while achieving their individual goals. This algorithm has found applications in various robotic domains, including autonomous vehicles, swarm robotics, and multi-robot systems in industrial settings. Its hierarchical and cooperative nature makes it well-suited for scenarios where multiple agents need to navigate in a coordinated manner while adhering to safety constraints.

### 4.3.Conflict-Based Search (CBS)

In 2005, Sharon et al. presented Conflict-Based Search (CBS), an algorithmic paradigm for resolving the multi-agent pathfinding (MAPF) problem. To increase its scalability and effectiveness, CBS has undergone various upgrades and changes since its debut.

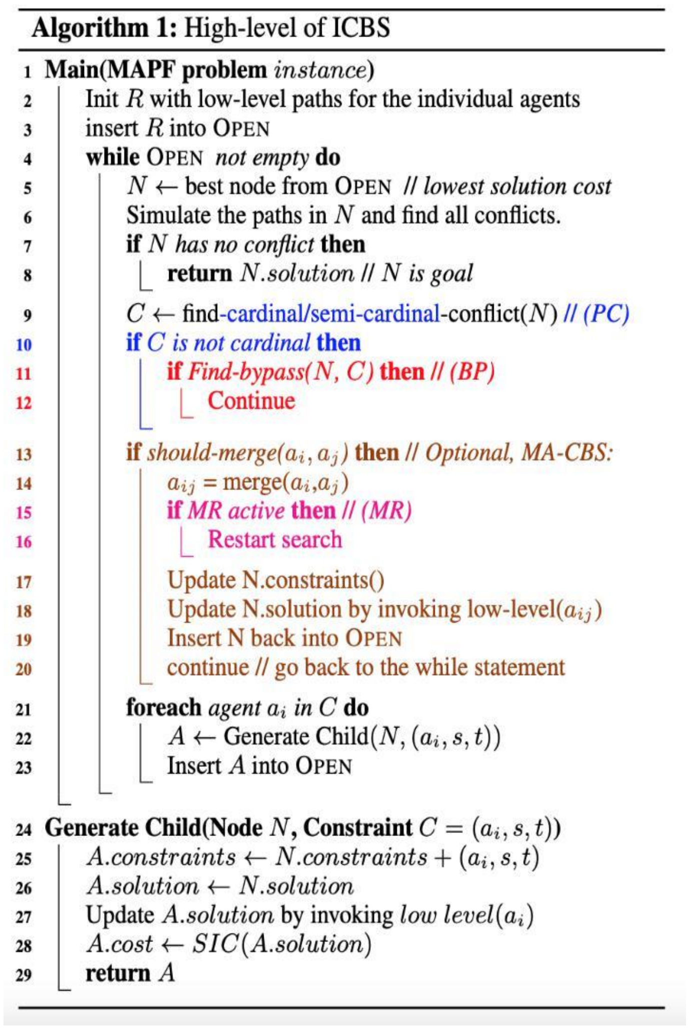
Prioritised Planning (PP), an early addition to CBS that was suggested by Silver in 2005, prioritizes the search's limits. PP enabled CBS to resolve more significant and intricate

MAPF issues. The addition of Independence Detection (ID), which was suggested by Sharon et al. in 2008, was another enhancement to CBS. To narrow the search space and improve the scalability of CBS, ID takes advantage of the independence amongst agents. Enhanced Conflict-Based Search (ECBS), which presented a new method for handling conflicts in the search, was proposed by Sharon et al. in 2009. The dispute resolution policy utilized by ECBS is more adaptable than the one employed by CBS, enabling more efficient searches in challenging circumstances. Later, in 2011, Boyarski and Sharon put out the Generalized-Screening CBS (GS-CBS) form of CBS, which makes use of a cutting-edge pruning method based on unfavorable solutions to increase scalability. Boyarski et al. presented the idea of incremental search to CBS in 2012. The process of incremental search entails breaking down the search space into smaller subproblems and addressing each one one at a time. CBS can manage more substantial and challenging issues because of this method. Since its debut, CBS has evolved significantly, and many changes and upgrades have been suggested to increase its scalability and effectiveness. These adjustments have made it possible for CBS to tackle larger and more intricate MAPF issues and have prompted the creation of several algorithmic variations that specifically target different difficulties in MAPF.

### 4.4.Improved Conflict-Based Search Algorithm for Multi-Agent Pathfinding(ICBS)

The research paper titled "ICBS: Improved Conflict-Based Search Algorithm for Multi-Agent Pathfinding" by Eli Boyarski proposes an improved version of the Conflict-Based Search (CBS) algorithm for Multi-Agent Pathfinding (MAPF) problems. Although the CBS algorithm is a popular approach for solving MAPF problems, it has some limitations regarding scalability and efficiency. The proposed ICBS algorithm addresses these limitations by introducing modifications such as the use of a more efficient priority queue data structure to store the search nodes, a novel heuristic function that considers the cost of future conflicts, and a dynamic threshold strategy to adaptively adjust the search space. The experimental results show that the ICBS algorithm outperforms the original CBS algorithm and other state-of-the-art MAPF algorithms in terms of solution quality and computation time on various benchmark instances. The ICBS algorithm also scales better with the number of agents and the size of the environment, making it suitable for solving large-scale MAPF problems. In summary, the ICBS algorithm is a significant improvement over the CBS algorithm for solving MAPF problems and is a valuable contribution to the field of multi-agent systems and robotics. One of the significant improvements of ICBS is the use of an efficient priority queue data structure that reduces the number of nodes requiring expansion, resulting in a significant reduction in computation time. The proposed algorithm's novel heuristic function helps to guide the search process more effectively by focusing on the most promising regions of the search space.

ICBS introduces a dynamic threshold strategy to adaptively adjust the search space, allowing the algorithm to focus on the most promising regions and ignore less favorable areas. This improves the scalability and reduces computation time. The research paper demonstrates through experimental results that ICBS surpasses the original CBS algorithm and other MAPF algorithms in terms of solution quality and computation time on a variety of benchmark instances. ICBS scales better with the number of agents and the environment size, making it suitable for large-scale MAPF problems. Consequently, the proposed ICBS algorithm is a significant improvement over the CBS algorithm, making it a valuable contribution to multi-agent systems and robotics. ICBS can be applied in various applications such as warehouse automation, traffic control, and unmanned aerial vehicle (UAV) coordination. The paper presents an enhanced version of ICBS by Eli Boyarski that addresses issues with the current implementation of ICBS, including cases where it fails to find a solution or takes a long time.



**Algorithm 1:** High-level of ICBS

```
Main(MAPF problem $instance$)
    Init $R$ with low-level paths for the individual agents
    insert $R$ into OPEN
    while OPEN not empty do
        $N \leftarrow$ best node from OPEN // lowest solution cost
        Simulate the paths in $N$ and find all conflicts.
        if $N$ has no conflict then
            return $N.solution$ // N is goal
        $C \leftarrow$ find-cardinal/semi-cardinal-conflict($N$) // (PC)
        if $C$ is not cardinal then
            if Find-bypass($N$, $C$) then // (BP)
                Continue
        if should-merge($a_i, a_j$) then // Optional, MA-CBS:
            $a_{ij}$ = merge($a_i$,$a_j$)
            if MR active then // (MR)
                Restart search
            Update N.constraints()
            Update N.solution by invoking low-level($a_{ij}$)
            Insert N back into OPEN
            continue // go back to the while statement
        foreach agent $a_i$ in $C$ do
            $A \leftarrow$ Generate Child($N, (a_i, s, t)$)
            Insert $A$ into OPEN
Generate Child(Node $N$, Constraint $C = (a_i, s, t)$)
    $A.constraints \leftarrow N.constraints + (a_i, s, t)$
    $A.solution \leftarrow N.solution$
    Update $A.solution$ by invoking $low\ level(a_i)$
    $A.cost \leftarrow SIC(A.solution)$
    return $A$
```

Multi-Agent Pathfinding (MAPF) is an important problem in the field of robotics and artificial intelligence, where multiple agents must navigate through a shared environment without colliding with each other. MAPF can be applied in various domains, such as warehouse automation, traffic control, and unmanned aerial vehicle coordination. The Improved Conflict-Based Search (ICBS) algorithm is a popular approach to solving MAPF problems.

However, ICBS has some limitations, such as being stuck in certain situations when there are many agents and few paths available. In a recent paper, Eli Boyarski proposed several improvements to the ICBS algorithm to address these limitations. One of the improvements proposed is prioritized planning, which involves planning agents in a certain order based on a heuristic. This helps to reduce the number of conflicts that need to be resolved, as agents planned first can avoid conflicts that later agents would encounter. Another improvement is independent conflict avoidance, which divides agents into groups and plans their routes independently. This can help to avoid conflicts that would be difficult to resolve otherwise.

The third improvement is the delayed conflict resolution strategy, which involves delaying the resolution of conflicts until they become necessary, rather than resolving them immediately.

This can help to reduce the number of conflicts that need to be resolved overall, as some conflicts may resolve themselves without needing explicit intervention. The fourth improvement is the expansion conflict detector, which detects conflicts during the expansion phase of the search process, rather than during the conflict detection phase. This helps to reduce the search space by avoiding unnecessary branches that lead to conflicts. To evaluate the effectiveness of the enhanced ICBS algorithm, Boyarski conducted a series of experiments on standard MAPF benchmarks. The results showed that the enhanced algorithm outperformed the original ICBS algorithm in terms of both solution quality and runtime in all cases. The largest improvement was observed in cases where there were many agents and few paths available, where the enhanced algorithm could find solutions up to five times faster than the original. In conclusion, Boyarski's proposed improvements to the ICBS algorithm are a significant contribution to the field of multi-agent systems and robotics. The enhanced ICBS algorithm is more efficient and effective in solving MAPF problems, making it suitable for large-scale applications.

### 4.5.Priority Inheritance with Backtracking ( PIBT)

PIBT works by assigning priorities to each agent based on their position in the environment or other criteria. The algorithm then attempts to move the agents along their optimal paths while avoiding collisions with other agents. If a collision is detected, PIBT employs a backtracking mechanism to temporarily suspend the lower-priority agent and allow the higher-priority agent to move.

The backtracking mechanism in PIBT involves temporarily suspending the lower-priority agent and attempting to move the higher-priority agent again. This process continues until all agents have reached their destinations without any collisions. The result is a provably correct and optimal solution to the MAPF problem.

PIBT has several advantages over other MAPF algorithms, including the ability to handle large and complex environments, the ability to generate provably correct and optimal solutions, and the ability to minimize communication overhead between agents. However, PIBT may not be suitable for scenarios where agents have incomplete or inconsistent knowledge of the environment, or where the robustness of the solution is a critical requirement. In summary, PIBT is a powerful algorithmic approach for solving MAPF problems that prioritize the generation of provably correct and optimal solutions, making it a suitable choice for scenarios where these criteria are critical requirements.

## 5. Comparison

### 5.1.A*, HCA* and WHCA*:

Several studies [5] have shown that when a series of challenging, maze-like environments consisting of 32x32 4-connected grids with 20% randomly placed impassable obstacles and filled disconnected subregions were tested with different algorithms, all methods achieved a 100% success rate with a small number of agents. However, when more agents were added to the map, Local Repair A* was not able to cope with the congestion, resulting in 20% of the agents failing to reach their destinations due to a bottleneck. On the other hand, the three Cooperative A* algorithms allowed agents to navigate through bottlenecks by stepping aside and allowing others to pass, resulting in a higher success rate even with a larger number of agents. Among these algorithms, WHCA* with a window size of 16 (indicated in parentheses in the figures) achieved the best success rate, with less than 2% of the agents failing to reach their destination, making it a more efficient and effective approach for pathfinding in crowded environments.

#### 5.1.1 How is WHCA better than HCA?

One issue with the previous algorithms is how they terminate once the agents reach their destination. If an agent sits on its destination, for example in a narrow corridor, then it may block off parts of the map to other agents. Ideally, agents should continue to cooperate after reaching their destinations, so that an agent can move off its destination and allow others to pass. A second issue is the sensitivity to agent ordering. Although it is sometimes possible to prioritise agents globally (Latombe 1991), a more robust solution is to dynamically vary the agent order, so that every agent will have the highest priority for a short period. Solutions can then be found which would be unsolvable with an arbitrary, fixed agent order. Thirdly, the previous algorithms must calculate a complete route to the destination in a large, three-dimensional state space. With single-agent searches, planning, and plan execution are often interleaved to achieve greater efficiency (see for example Korf's Real-Time Heuristic Search (1990)), by avoiding the need to plan for long-term contingencies that do not occur. WHCA* develops a similar idea for cooperative search. A simple solution to all of these issues is to window the search. The cooperative search is limited to a fixed depth specified by the current window. Each agent searches for a partial route to its destination and then begins following the route. At regular intervals (e.g. when an agent is halfway through its partial route) the window is shifted forwards and a new partial route is computed. To ensure that the agent heads in the correct direction, only the cooperative search depth is limited to a fixed depth, whilst the abstract search is executed to full depth. A window of size w can be viewed as an intermediate abstraction that is equivalent to the base level state space for w steps, and then equivalent to the abstract level state space for the remainder of the search. In other words, other agents are only considered for w steps (via the reservation table) and are ignored for the remainder of the search. To search this new search space efficiently, a simple trick can be used. Once w steps have elapsed, agents are ignored and the search space becomes identical to the abstract search space. This means that the abstract distance provides the same information as completing the search. For each node Ni reached after w steps a special terminal edge is introduced, going directly from Ni to the destination G, with a cost equal to the abstract distance from Ni to G. Using this trick, the search is reduced to a w-step window using the abstract distance heuristic introduced for HCA*.

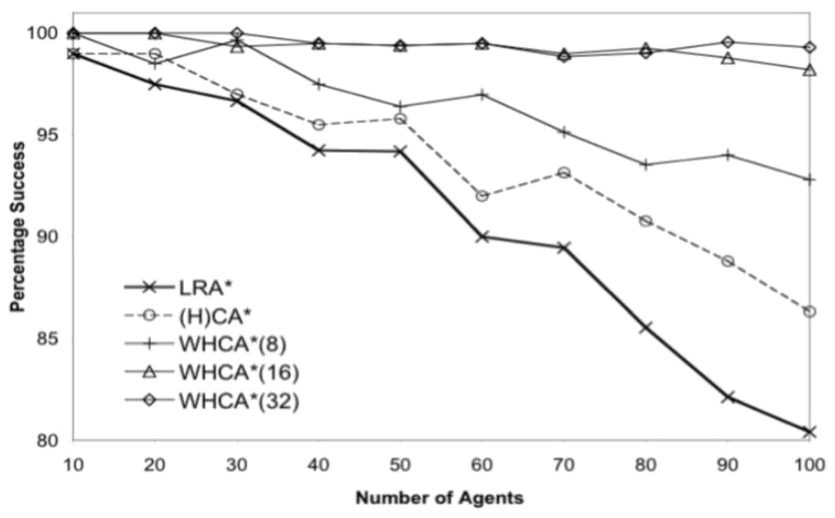


**Fig 5.1.1** Comparison of WHCA , LRA ,H(CA)*

In addition, the windowed search can continue once the agent has reached its destination. The agent's goal is no longer to reach the destination but to complete the window via a terminal edge. Any sequence of w moves will thus reach the goal. However, the WHCA* search will efficiently find the lowest cost sequence. This optimal sequence represents the partial route that will take the agent closest to its destination, and once there stay on the destination for as much time as possible Experiment result shows that WHCA takes 9 times to complete the path which is more than less than HCA but more than PIBT. It has less Sum of cost is more in WHCA compared to HCA

## 5.2.Comparison between PIBT, CBS, and ICBS

One of the key advantages of PIBT over CBS and ICBS is its ability to handle multiple types of constraints, such as temporal and energy constraints, in addition to the traditional collision avoidance constraints. This is due to its ability to use a priority function that takes into account multiple criteria, allowing for more flexible and efficient path planning.

Another advantage of PIBT is its ability to handle high levels of uncertainty in the environment, such as incomplete information about other agents' goals or movements. PIBT can handle this uncertainty by using a backtracking mechanism that allows agents to revise their plans when new information becomes available. In terms of computational complexity, PIBT has a worst-case time complexity of O(n^3), where n is the number of agents, while CBS and ICBS have a worst-case time complexity of O(b^n), where b is the branching factor of the search tree. This makes PIBT a more scalable and efficient approach for large-scale multi-agent pathfinding problems. Finally, PIBT has been shown to outperform CBS and ICBS in several benchmarks and real-world scenarios, including scenarios with dynamic obstacles and multiple types of constraints.

## 5.3.Comparison between PIBT, HCA*, and WHCA*

Compared to HCA* and WHCA*, PIBT assumes that agents have complete and consistent knowledge of the environment, which allows it to generate optimal solutions that are provably correct. In contrast, HCA* and WHCA* assume that agents have incomplete and inconsistent knowledge, which makes it challenging to guarantee optimality or correctness.

Moreover, PIBT can handle complex planning problems where agents have conflicting goals, as it can resolve conflicts by assigning priorities based on the goals of the agents involved.

This is not the case with HCA*, which generates subtasks for agents without considering their goals or priorities, potentially leading to suboptimal or inefficient solutions. Additionally, PIBT can generate solutions that are not only optimal but also robust to changes in the environment, while HCA* and WHCA* may require significant replanning when the environment changes.

However, PIBT is computationally expensive, especially for problems with a large number of agents or complex temporal constraints. To improve the algorithm's efficiency, PIBT can be optimized using techniques such as heuristic search and pruning. Overall, PIBT is a suitable choice for problems that require optimal solutions and intricate planning, but may not be the best choice when there is a need for agents to cooperate and share information to achieve their goals, or when the environment is subject to frequent changes that require dynamic replanning.

### 5.4.Comparison between ICBS and PIBT:

The Improved Conflict-Based Search Algorithm for Multi-Agent Pathfinding (ICBS) and Priority Inheritance with Backtracking (PIBT) are two popular approaches for solving the Multi-Agent Pathfinding (MAPF) problem. While ICBS is an improved version of the Conflict-Based Search (CBS) algorithm, PIBT is a method that uses priority inheritance and backtracking to solve MAPF problems. Both ICBS and PIBT are effective in solving a variety of MAPF problems. ICBS is effective in finding high-quality solutions quickly, while PIBT is effective in solving problems with large numbers of agents.

In terms of solution quality, ICBS has been shown to outperform the original CBS algorithm and other state-of-the-art MAPF algorithms, including PIBT, on a variety of benchmark instances. However, PIBT is effective in solving large-scale MAPF problems, whereas ICBS may not be as effective due to its high computational requirements. One key difference between ICBS and PIBT is the use of priority inheritance in PIBT. This technique allows agents with higher priority to preempt agents with lower priority to avoid conflicts. This can be particularly effective in situations where there are many agents with similar start and goal locations. In contrast, ICBS uses a conflict-based approach that focuses on resolving conflicts between agents by delaying their movements and assigning higher priority to agents that are involved in more conflicts.

Another key difference between ICBS and PIBT is the use of backtracking in PIBT. This technique allows agents to backtrack and re-plan their paths to avoid conflicts. This can be particularly effective in situations where agents need to make decisions based on uncertain or changing information. In contrast, ICBS uses a heuristic function to guide the search and prioritize the search space based on the likelihood of finding a solution. In summary, the choice between ICBS and PIBT depends on the specific requirements of the MAPF problem at hand. If finding high-quality solutions quickly is the priority, ICBS may be a better choice. If solving problems with large numbers of agents or uncertain information is the priority, PIBT may be a better choice due to its use of priority inheritance and backtracking techniques.

### 5.5.Comparison between ICBS and WHCA

Two well-liked methods for resolving the multi-agent pathfinding (MAPF) problem are Windowed Hierarchical Cooperative A* (WHCA*) and Improved Conflict-Based Search

Algorithm for Multi-Agent Pathfinding (ICBS). The Conflict-Based Search (CBS) method has been enhanced, and the ICBS algorithm is intended to identify high-quality solutions for a variety of MAPF problems efficiently. In contrast to CBS, the algorithm makes several changes that improve its scalability and efficiency. These changes include the introduction of a novel heuristic function, a more effective priority queue data structure, and a dynamic threshold technique.

On the other hand, WHCA* is a centralised method that addresses MAPF issues via a hierarchical cooperative search technique. The method breaks the issue down into more manageable subproblems, which are then collectively solved hierarchically. To detect potential collisions and avoid them in real-time, WHCA* also employs a window-based method. On a variety of benchmark situations, it has been demonstrated that ICBS performs better than the original CBS algorithm and other cutting-edge MAPF algorithms in terms of solution quality and computation time. Additionally, it has been demonstrated that ICBS scales better with the number of agents and the size of the environment, making it appropriate for handling complex MAPF issues.

It has been demonstrated that WHCA* works well in real-world circumstances and can identify workable solutions for a large number of agents quickly. However, the quality of the solutions might not always be the best, and the algorithm might have trouble handling more complicated issues. A 2017 research titled "Comparing Cooperative Pathfinding Algorithms: A Study on Large-Scale Multi-Agent Systems" conducted a comparison between ICBS and WHCA*. On a range of benchmark examples, the study discovered that ICBS beat WHCA* in terms of solution quality, scalability, and computational efficiency.

## 6. Conclusion

The iterative improvement of path finding for several robots was given in this paper. The idea employs two MAPF solvers as sub-procedures: an imperfect solver to come up with a preliminary solution and an ideal solver to hone it. The framework discovers a solution with acceptable costs in a given situation, although this does not ensure finding the optimal answer in quick computation time that is highly scalable. Additionally, it has time planning, which is a desirable feature for real-time systems with strict deadlines. The studies show that regardless of the starting solutions, the cost is lowered to almost optimal levels; nonetheless, it is preferable to start with solutions that are at least adequate in terms of efficiency because we may develop better solutions sooner. Therefore, the following whenever the MAPF method will be useful. Start numerous initial solvers in parallel with varying runtime and solution quality portfolios. After that, refine the initial answer you came up with. If another initial solver gets a better solution compared to the refined solution at that time, replace the current one with the new one.

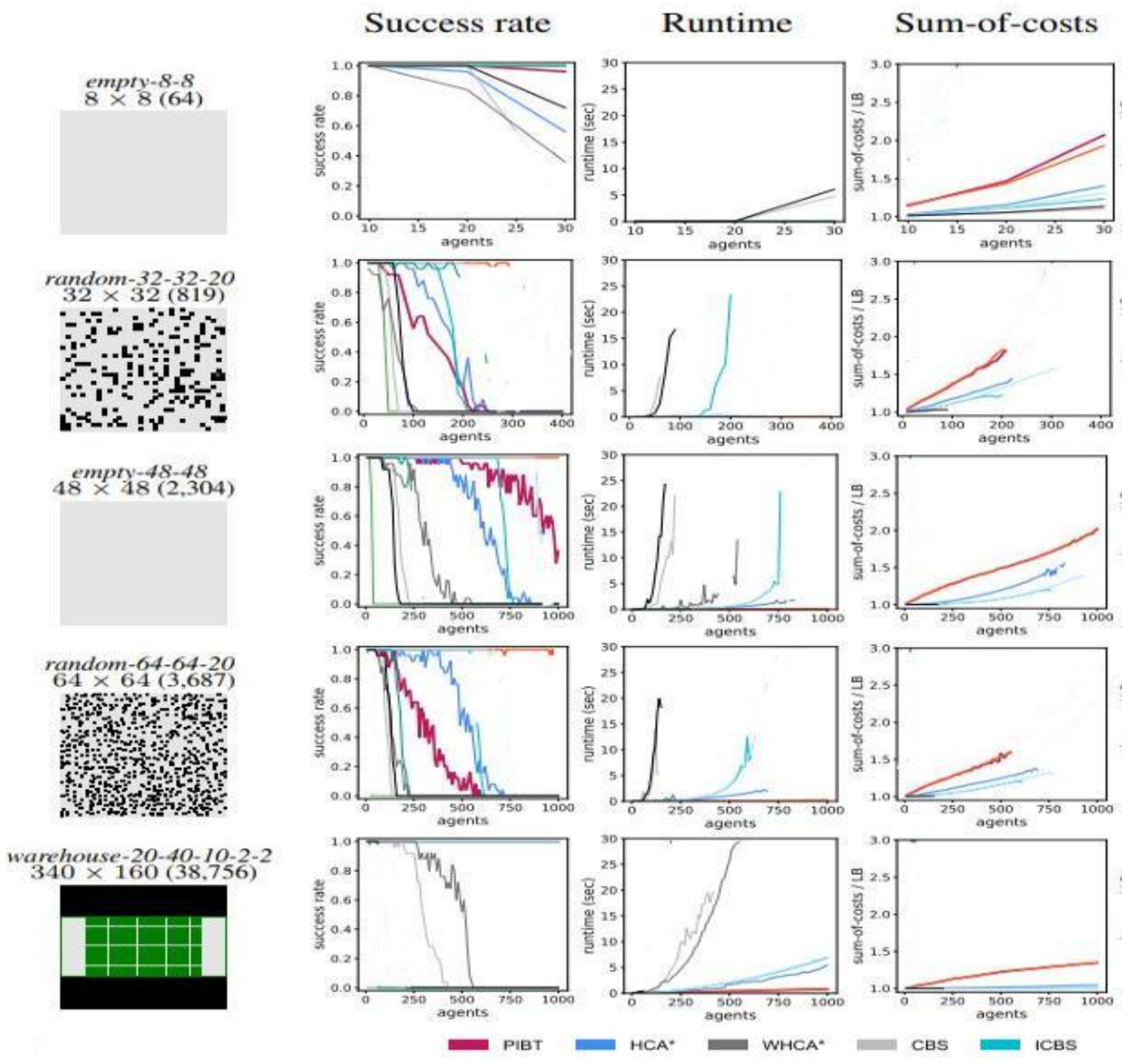


**Fig 6.1:** Comparison of Various Algorithms

References

[1] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," AI Magazine, vol. 29, no. 1, p. 9, 2008.

[2] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," J. Artif. Intell. Res., vol. 31, pp. 591–656, 2008.

[3] R. Morris, C. S. Pasareanu, K. S. Luckow, W. Malik, H. Ma, T. S. Kumar, and S. Koenig, "Planning, scheduling and monitoring for airport surface operations." in AAAI Workshop: Planning for Hybrid Systems, 2016.

[4] A. Okoso, K. Otaki, and T. Nishi, "Multi-agent path finding with priority for cooperative automated valet parking," in Proc. IEEE Intell. Transp. Syst. Conf. (ITSC), 2019, pp. 2135–2140.

[5] D. Silver, "Cooperative pathfinding." AIIDE, pp. 117–122, 2005.

[6] P. Surynek, "A novel approach to path planning for multiple robots in bi-connected graphs," in Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), 2009, pp. 3613–3619.

[7] K.-H. C. Wang and A. Botea, “Mapp: a scalable multi-agent path planning algorithm with tractability and completeness guarantees,” J. Artif. Intell. Res., vol. 42, pp. 55–90, 2011.

[8] R. Luna and K. E. Bekris, “Push and swap: Fast cooperative pathfinding with completeness guarantees,” in Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI), 2011, pp. 294–300.

[9] B. de Wilde, A. W. ter Mors, and C. Witteveen, “Push and rotate: cooperative multi-agent path planning,” in Proc. Int. Joint Conf. on Autonomous Agents & Multiagent Systems (AAMAS), 2013, pp. 87–94.

[10] K. Okumura, M. Machida, X. Defago, and Y. Tamura, “Priority ´ inheritance with backtracking for iterative multi-agent pathfinding,” in Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI), 2019, pp. 535–542.

[11] S. Zilberstein, “Using anytime algorithms in intelligent systems,” AI Magazine, vol. 17, no. 3, pp. 73–73, 1996.

[12] G. Wagner, Subdimensional expansion: A framework for computationally tractable multi-robot path planning (Jul 2018). doi:10.1184/R1/6723329.v1.

[13] G. Sharon, R. Stern, A. Felner, N. R. Sturtevant, Conflict-based search for optimal multi-agent pathfinding, Artificial Intelligence 219 (2015) 40–66.

[14] K. Vedder, J. Biswas, X*: Anytime multiagent planning with bounded search, in Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS ’19, International 24 Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2019, p. 22472249.

[15] R. Luna, K. E. Bekris, Push and swap: Fast cooperative path-finding with completeness guarantees, in Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One, IJCAI’11, AAAI Press, 2011, pp. 294–300. doi:10.5591/ 978-1-57735-516-8/IJCAI11-059. URL http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-059

[16] J. Yu, Average case constant factor time and distance optimal multi-robot path planning in well-connected environments, Autonomous Robots 44 (2020) 469–483.URL https://doi.org/10.1007/s10514-019-09858-z

[17] D. Silver, Cooperative pathfinding, in Proceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE’05, AAAI Press, 2005, p. 117122.

[18] V. R. Desaraju, J. P. How Decentralized path planning for multi-agent teams with complex constraints, Autonomous Robots 32 (2012) 385–403. URL https://doi.org/10.1007/s10514-012-9275-2

[19] S. S. Chouhan, R. Niyogi, Dmapp: A distributed multi-agent path planning algorithm, in Australasian Joint Conference on Artificial Intelligence, Springer, 2015, pp. 123–135.

[20] P. Pianpak, T. C. Son, Z. O. Toups, W. Yeoh, A distributed solver for multiagent pathfinding problems, in Proceedings of the First International Conference on Distributed Artificial Intelligence, 2019, pp. 1–7.

[21] A. Gilboa, A. Meisels, A. Felner, Distributed navigation in an unknown physical environment, in Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, 2006, pp. 553–560.

[22] H. Ma, J. Li, T. K. S. Kumar, S. Koenig, Lifelong multi-agent pathfinding for online pickup and delivery tasks, in Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, S˜ao Paulo, Brazil, May 8-12, 2017, 2017, pp. 837–845.

[23] M. Barer, G. Sharon, R. Stern, and A. Felner. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem.

[24]. E. Boyarski, A. Felner, G. Sharon, and R. Stern, 2015.:Bypassing conflicts in multi-agent pathfinding.

[25] In ICAPS-2015, Jerusalem, Israel, June 7-11, 2015. [Erdem et al., 2013] Esra Erdem, Doga G. Kisa, Umut Oztok, and Peter Schueller. A general formal framework for pathfinding problems with multiple agents.

[26] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar, et al., “Multi-agent pathfinding: Definitions, variants, and benchmarks,” in Proc. Int. Symp. on Combinatorial Search (SoCS), 2019, pp. 151–159.

[27] P. R. Wurman, R. D’Andrea, and M. Mountz, “Coordinating hundreds of cooperative, autonomous vehicles in warehouses,” AI Magazine, vol. 29, no. 1, p. 9, 2008.

[28] K. Dresner and P. Stone, “A multiagent approach to autonomous intersection management,” J. Artif. Intell. Res., vol. 31, pp. 591–656, 2008.

[29] R. Morris, C. S. Pasareanu, K. S. Luckow, W. Malik, H. Ma, T. S. Kumar, and S. Koenig, “Planning, scheduling and monitoring for airport surface operations.” in AAAI Workshop: Planning for Hybrid Systems, 2016.

[30] A. Okoso, K. Otaki, and T. Nishi, “Multi-agent path finding with priority for cooperative automated valet parking,” in Proc. IEEE Intell. Transp. Syst. Conf. (ITSC), 2019, pp. 2135–2140

[31] Marika Ivanov´a and Pavel Surynek. Adversarial cooperative path-finding: A first view. In *AAAI (Late-Breaking Developments)*, 2013.

[32] Qandeel Sajid, Ryan Luna, and Kostas E Bekris. Multi-agent pathfinding with simultaneous execution of single-agent primitives. In *SOCS*, 2012.

[33] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[34] Meir Goldenberg, Ariel Felner, Roni Stern, Guni Sharon, and Jonathan Scha effer. A* variants for optimal multi-agent pathfinding. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[35] Alexander Zelinsky. A mobile robot exploration algorithm. *IEEE Transactions on Robotics and Automation*, 8(6):707–717, 1992.

[36] Robert C Holte, Maria B Perez, Robert M Zimmer, and Alan J MacDonald. Hierarchical a*: Searching abstraction hierarchies efficiently. In *AAAI/IAAI, Vol. 1*, pages 530–535. Citeseer, 1996.

[37] Trevor Scott Standley. Finding optimal solutions to cooperative pathfinding problems. In *AAAI*, volume 1, pages 28–29, 2010.

[38] A. Felner, M. Goldenberg, R. Stern, G. Sharon, T. Beja, R. Holte, J. Schaeffer, N. Sturtevant, and Z. Zhang. Partial-expansion A* with selective node generation.AAAI, 2012.

[39] M. Goldenberg, A. Felner, R. Stern, G. Sharon, N. R. Sturtevant, R. C. Holte, and J. Schaeffer. Enhanced partial expansion A. J. Artif. Intell. Res. (JAIR), 50:141–187, 2014.

[40] G. Sharon, R. Stern, M. Goldenberg, and A. Felner. Pruning techniques for the increasing cost tree search for optimal multi-agent pathfinding. In SOCS, pages 150–157, 2011.

[41] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. In AAAI, 2012. [Sharon et al., 2012b] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant. Meta-agent conflict-based search for optimal multi-agent pathfinding. In SOCS, 2012.

[42] G. Sharon, R. Stern, M. Goldenberg, and Ariel Felner. The increasing cost tree search for optimal multi-agent pathfinding. Artif. Intell., 195:470–495, 2013.

[43] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. Artificial Intelligence Journal (AIJ), 218:40–66, 2015.

[44] T. S. Standley. Finding optimal solutions to cooperative pathfinding problems. In AAAI, 2010.

[45] Roni Stern, Tamar Kulberis, Ariel Felner, and Robert Holte. Using lookaheads with optimal best first search. In AAAI, 2010.