

# An Analysis of Constraint-Based Multi-Agent Pathfinding Algorithms

Hannah Lee<sup>1</sup>, James D. Motes<sup>1</sup>, Marco Morales<sup>1,2</sup>, and Nancy M. Amato<sup>1</sup>

**Abstract**—This study informs the design of future multi-agent pathfinding (MAPF) and multi-robot motion planning (MRMP) algorithms by guiding choices based on constraint classification for constraint-based search algorithms. We categorize constraints as conservative or aggressive and provide insights into their search behavior, focusing specifically on vanilla Conflict-Based Search (CBS) and Conflict-Based Search with Priorities (CBSw/P). Under a hybrid grid-roadmap representation with varying resolution, we observe that aggressive (priority constraint) formulations tend to solve more instances as agent count or resolution increases, whereas conservative (motion constraint) formulations yield stronger solution quality when both succeed. Findings are synthesized in a decision flowchart, aiding users in selecting suitable constraints. Recommendations extend to Multi-Robot Motion Planning (MRMP), emphasizing the importance of considering topological features alongside problem, solution, and representation features. A comprehensive exploration of the study, including raw data and map performance, is available in our public GitHub Repository<sup>3</sup>.

**Index Terms**—Path Planning for Multiple Mobile Robots or Agents, Motion and Path Planning

## I. INTRODUCTION

**M**ULTI-AGENT Pathfinding (MAPF) is a critical multi-agent coordination problem wherein safe and efficient paths are planned for multiple agents. MAPF finds applications in domains such as assembly [1]–[3], evacuation [4], formation control [5]–[9], localization [10], and object transportation [3], [11]. The core challenge lies in orchestrating the concurrent traversal of agents along their respective paths without colliding with each other or with environmental obstacles. MAPF is an NP-hard problem that exhibits an exponentially growing state space with the number of agents [12].

A widely adopted strategy for addressing MAPF involves using constraint-based search algorithms, which iteratively constrain the search space to identify valid solutions. Many existing algorithms, including those presented in [13]–[16], can be generalized within this constraint-based search paradigm. In this paper, we specifically examine two prominent algorithms: Conflict-Based Search (CBS) [17] and Conflict-Based Search with Priorities (CBSw/P) [18]. CBS independently plans paths for individual agents, resolving conflicts by adding constraints on specific agent actions at given timesteps. CBSw/P, a variant of CBS employing prioritized planning, resolves conflicts by assigning priorities to agents, thus constraining lower-priority agents based on higher-priority ones.

<sup>1</sup>Hannah Lee, James D. Motes, Marco Morales, and Nancy M. Amato are with the Parasol Lab, School of Computer Science, University of Illinois at Urbana Champaign, Champaign, IL, 61820 USA. hannah9, jmot2, moralesa, namato@illinois.edu

<sup>2</sup>Marco Morales is also with the Department of Computer Science at Instituto Tecnológico Autónomo de México (ITAM), Mexico City, México.

<sup>3</sup><https://github.com/hannahjmlee/constraint-mapf-analysis>

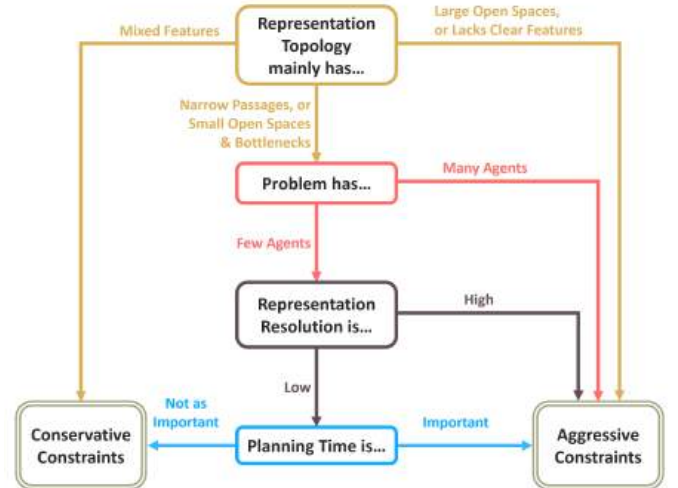


Fig. 1. A flowchart summarizing our findings and providing general set guidelines for determining when to use aggressive vs. conservative constraints given our heuristics derived from comparing CBS and CBSw/P. These decision points may shift with other search strategies or under different settings.

To effectively evaluate the performance of constraint-based search algorithms, it is essential to assess and characterize problem difficulty. Problem difficulty directly influences the efficiency of an algorithm’s conflict resolution strategy, affecting both computational complexity and solution quality. The choice of constraints significantly impacts the shape and size of the search space and is highly sensitive to problem difficulty. However, measuring the difficulty of a specific MAPF instance remains challenging due to the dynamic nature of conflict resolution in multi-agent systems. Traditional analyses estimate problem difficulty using factors such as problem size, environment topology, and environment dimensions, but these approaches do not generalize well to more complex representations, such as the roadmaps commonly used in multi-robot motion planning. This work investigates how different constraint choices affect search behavior across multiple dimensions of problem difficulty, emphasizing the importance of selecting appropriate constraints for both MAPF and more general representations.

To address this need, we introduce a framework classifying constraints as conservative (akin to CBS) or aggressive (akin to CBSw/P). Through CBS and CBSw/P, we investigate the implications of using conservative versus aggressive constraints and analyze performance variations relative to representation topology and resolution. Our findings provide insights for informed decision-making in constraint-based search algorithm design and offer critical considerations for adapting MAPF solvers to more complex domains, such as Multi-Robot Motion Planning (MRMP). We summarize our contributions

and insights in the flowchart depicted in Figure 1.

### A. MAPF vs. MRMP

Past studies of MAPF algorithms have concentrated mainly on the challenges presented by different environmental topologies, such as narrow passages and open areas [17]–[20]. However, these studies do not directly apply to MRMP due to a key difference in how each approaches representation. In MAPF scenarios, the emphasis is on searching within an already established representation for a team plan. In contrast, MRMP can involve generating or implicitly defining a representation that captures the robot’s geometry, kinematics, and allowable motions. For instance, many sampling-based methods construct a roadmap approximating the free configuration space, whereas search-based methods can define an implicit graph through motion primitives [21]. In either case, MRMP requires more detailed modeling than MAPF. It is important to note that our paper does not explore the construction of these representations; instead, we focus on evaluating how MAPF querying techniques perform across a range of representations.

MAPF usually coordinates the movements of several agents by mapping their states onto a uniform grid, effectively simplifying the robot’s details. On the other hand, MRMP considers the robot’s shape, size, and orientation, often relying on sampling-based methods to generate non-uniform roadmaps or leveraging implicit graphs that approximate the connectivity of the free configuration space. Therefore, MRMP demands a more detailed representation of robots and their configuration spaces while leveraging MAPF algorithms for coordinating their motions on the generated representations.

The standard uniform grid representation of MAPF generates a grid map that simplifies collision checking through atomic actions and edges. Thus, collision checking in MAPF is typically simple and efficient. In contrast, the non-uniform roadmap representation of MRMP is characterized by diverse edge lengths and durations. These pose collision checking challenges distinct from traditional MAPF scenarios because the robot’s shape, volume, and orientation must be considered. Additionally, the variability in edge length and resolution in a non-uniform representation can give rise to artificial graph topology.

As illustrated in the top set of images in Figure 2, in an environment devoid of narrow passages, sparse roadmaps with long edges can create artificial narrow passages during the search. These narrow passages, absent in the actual environment topology, emerge as a result of the representation during the search. Conversely, as illustrated in the bottom set of images in Figure 2, in environments with narrow passages, the representation may not capture them and result in a roadmap that lacks such passages during the search. This dynamic emphasizes the significant influence of representation topology on MRMP. The complexities arising from the varied resolutions and edge lengths underscore the challenges of directly applying MAPF methodologies to MRMP, particularly when relying solely on environmental topological analysis.

In our analysis, we use a grid roadmap representation to model the MAPF problem using roadmaps. This distinction

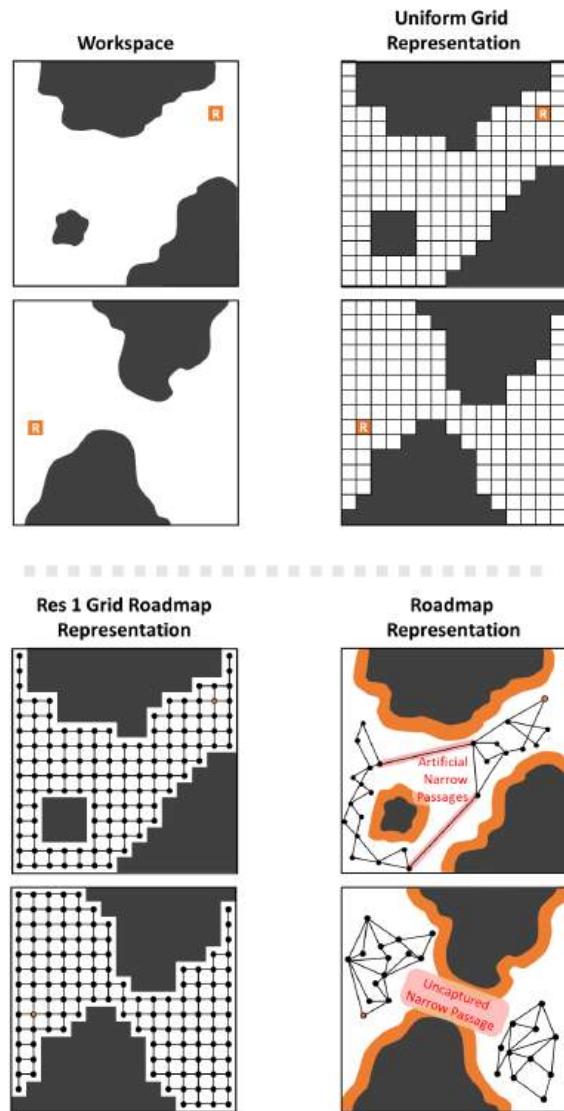


Fig. 2. Comparison of Grid, Grid Roadmap, and Roadmap Representations for a 2D square robot with 2 DoF. The orange position  $R$  represents the robot across all three representations. In the grid representation, states are mapped to discrete cells. The grid roadmap uses a uniform grid-sampler to map states to configurations, while the roadmap representation employs a random sampler to map states to the configuration space. For the roadmap, obstacles (gray) are inflated by the robot’s radius (orange) to define the obstacle space. The top row depicts an environment without narrow passages, where random sampling inadvertently creates one. In contrast, the bottom row shows a narrow passage environment that the roadmap fails to capture, resulting in a representation that overlooks the passage.

is shown in Figures 2 and 3 and is discussed in detail in Section IV-A. To capture the characteristics of both MAPF grids and MRMP roadmaps, we employ a hybrid grid roadmap representation. To clarify, our focus is not on the construction of representations; hence, we employ a simplified hybrid representation that captures the qualities of both grid and roadmap representations.

At a resolution of 1, the grid roadmap representation shows the same behavior as seen in traditional MAPF. As the resolution increases, we maintain the grid structure of MAPF, but mimic the behavior of denser roadmap representations encountered in MRMP. Our hybrid representation

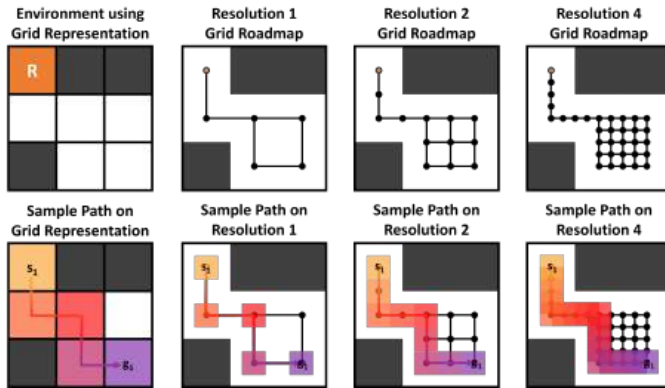


Fig. 3. Shown are samples of true grid and grid roadmap representations at resolutions 1, 2, and 4. This highlights cell-based planning in true grids and vertex-based planning in grid roadmaps. In grid roadmaps, resolution 1 grids equate to 4 vertices, while resolution 2 and 4 grids scale to 9 and 25 vertices, respectively. Sample paths are depicted for each resolution, showcasing increased vertex count with higher resolutions. When projecting a resolution 1 path into higher resolutions, the number of states in the path increases, akin to increasing edge discretization in a MRMP roadmap representation.

allows us to navigate the complexities of MRMP roadmap representations while retaining control over the granularity of representation resolution. This simplified representation mitigates the possibility of an inadequate or problematic (e.g., artificial narrow passages) representation of the workspace affecting our analysis of the search process. Therefore, we operate under the assumption that we are working with a well-constructed representation, allowing us to focus solely on the comparative performance of MAPF techniques across different representations.

### B. Key Takeaways

In this paper, we define aggressive and conservative constraints as a foundation for our discussion, supported by thorough experiments. We examine how these constraints affect various elements of constraint-based search algorithms. Our advice is presented in a user-friendly decision flowchart, shown in Figure 1, facilitating the selection of aggressive or conservative constraints for constraint-based search algorithms. Conservative constraints meticulously constrain the state space, prioritizing incremental progress towards the optimal solution while minimizing their impact during the search. In contrast, aggressive constraints aim to advance rapidly in the search space, with a broader influence that emphasizes efficient exploration.

Deciding between aggressive and conservative constraints for a MAPF or MRMP problem involves assessing various factors, each represented by a distinct color within our flowchart: the environment representation’s topology (tan), problem size (pink), representation resolution (purple), and desired planning time (blue). By systematically considering these factors, practitioners can tailor their constraint selection to align with the specific requirements and characteristics of the problem environment, ensuring optimal performance and efficient problem-solving.

- 1) *Environment Representation Characteristics (tan)*: Start by evaluating the topological features within your en-

vironment representation. If the topology consists of predominantly large open spaces or lacks clear features, one should opt for aggressive constraints. Environments with a mix of different features should use conservative constraints. However, if comprised of small open spaces and bottlenecks or narrow passages, move on to evaluate problem size and representation resolution.

- 2) *Problem Size (pink) and Representation Resolution (purple)*: For scenarios with many agents or high representation resolution, aggressive constraints can improve runtime efficiency and increase the chances of finding a solution before computational resources are exhausted. However, this comes at the cost of completeness, as more aggressive constraints may exclude valid solutions. Conversely, for smaller problem sizes and lower representation resolutions, the choice of constraints should be guided by the desired balance between solution completeness and computational performance.
- 3) *Performance Expectations (blue)*: Lastly, we consider the available planning time. If computational resources and runtime are not constraints, conservative constraints are a suitable choice. However, when planning time is limited, employing aggressive constraints can improve efficiency at the potential cost of completeness.

### C. Outline and Recommendations for Readers

This work is intended to serve two primary audiences: (1) readers already familiar with constraint-based search algorithms and their applications to multi-agent systems, and (2) those seeking a deeper foundational understanding of these algorithms.

For readers already familiar with constraint-based search algorithms, multi-agent pathfinding, and multi-robot motion planning, we recommend proceeding directly to Sections III-B, IV, and V. These sections classify constraints based on their impact on search behavior, present techniques for selecting constraints, and analyze the role of representation topology, including methods for estimating problem difficulty. Sections III-B and V offer insights into improving algorithm design, while Section V validates our findings through extensive experimental evaluation.

For readers seeking a more comprehensive and foundational understanding, we encourage reviewing all sections, as they establish the necessary context for our discussions. Sections II and III provide a comprehensive introduction to multi-agent pathfinding, multi-robot motion planning, and constraint-based search algorithms, ensuring a solid grasp of the foundational concepts. This work examines the search properties of constraints using the most basic version of constraint-based search algorithms to provide a clear and controlled analysis. Section III-A highlights recent advancements that improve scalability for large MAPF problems and extend applications beyond MAPF, offering new practitioners a starting point for exploring constraint-based search algorithms. Section IV details our methodology for evaluating and estimating problem difficulty, while Section V-B presents and analyzes our experimental results, reinforcing the key claims made throughout this work.

#### D. Scope and Limitations

This study characterizes how constraint formulation influences behavior by comparing vanilla CBS (motion constraints) with CBSw/P (priority constraints) under the hybrid grid-roadmap discretizations evaluated in Section V. Our goal is to provide controlled intuition about constraint families, not to exhaustively evaluate search strategies. Accordingly, we do not isolate or vary the high-level search or its heuristics (e.g., depth-first search PBS [18], bounded-suboptimal searches [22], heuristic-augmented searches [23], or LNS-style improvements [24]), and we do not claim that results for “conservative vs. aggressive” constraints generalize across those recent variants. The guidance offered here is most reliable for the vanilla versions and settings we study and should be treated as a first-pass heuristic when configuring newer methods. When contemporary high-level search algorithms, heuristics, or recent enhancements are employed, the observed trade-offs may shift.

In summary, our conclusions are scoped to vanilla CBS and CBSw/P using best-first searches under the representations and evaluation protocol described herein, and are not intended as general prescriptions for all conservative versus aggressive methods.

## II. PROBLEM DEFINITION

In our discussion of the classical MAPF problem, we will be referring to the problem described in [20], albeit with some variations in notation for simplicity and to maintain consistency with prior research. It is important to note that several MAPF variants exist, each contingent on specific assumptions about the problem definition. These variants address scenarios such as distributed settings, weighted graphs, feasibility constraints, large geometric or volumetric agents, and kinematic constraints [20].

In the classical MAPF problem [20], we are given an input consisting of  $n$  agents  $a_1, a_2, \dots, a_n$ . Each agent  $a_i$  is characterized by a start position  $s_i$  and goal position  $g_i$ . A path for an agent is denoted as  $\pi_i$  and is defined as the sequence of actions leading from its starting position  $s_i$  to its designated goal position  $g_i$ . The primary objective is to devise conflict-free paths for all  $n$  agents within a graph  $G$ .

The graph  $G = (V, E)$  is an undirected graph. Vertices  $V$  represent valid positions within the environment, while edges  $E$  signify permissible actions with duration to transition from one vertex to an adjacent one. Agents, at each timestep, are located on a graph vertex  $v \in V$ . They can opt to move along an edge  $(v, v') \in E$  or remain stationary at the current vertex  $v$ . For our grid roadmap representation, we assume a 4-neighbor movement model, where adjacent vertices are positioned directly up, down, right, and left of the current vertex. Similar to that of a true MAPF grid representation, the grid roadmap representation has atomic edges with a duration of one timestep per edge.

The graph’s representation resolution defines the granularity in the discretization of the configuration space. Higher resolution results in a more detailed and precise representation, enabling a clearer understanding of the agents’ potential states

and interactions. This is achieved through a finer grid, where sampling is conducted in a grid pattern with uniform spacing between points, as illustrated in Figure 3.

The MAPF solver aims to generate a team plan  $\pi = \{\pi_1, \dots, \pi_n\}$  comprising each agent’s path  $\pi_i$ . A path is represented as a sequence of vertices, denoted as  $\pi_i = (s_i, v, \dots, v', g_i)$ . The position of an agent  $a_i$  at timestep  $t$  is indicated as  $\pi_i[t]$ , corresponding to the  $t$ th vertex within  $\pi_i$ .

#### A. Conflicts

In MAPF, the objective is to devise a team plan that ensures the absence of conflicts among agents. A team plan is deemed valid if no conflicts exist between any pair of agents. For the classical MAPF problem, two types of conflicts are defined: *vertex conflict* and *edge conflict*. These conflicts are established based on the paths of a pair of agents  $a_i$  and  $a_j$ .

A *vertex conflict* between a pair of agent paths arises when both agents occupy the same vertex at the same time. Consequently, a vertex conflict between agents  $a_i$  and  $a_j$  at timestep  $t$  indicates the simultaneous presence of both agents at vertex  $v$ . This conflict is represented using a tuple  $\langle a_i, a_j, v, t \rangle$ .

An *edge conflict* arises when a pair of agents attempts to traverse the same edge simultaneously, regardless of the direction. Specifically, for agents  $a_i$  and  $a_j$  at timestep  $t$ , an edge conflict indicates their simultaneous attempt to traverse edge  $(v, v')$  between timesteps  $t$  and  $t + 1$ . This traversal can occur in either direction. When the pair of agents traverses the same edge in opposite directions, it is also known as a swapping conflict.

Edge conflicts are denoted using a tuple  $\langle a_i, a_j, (v, v'), t \rangle$ . Given that  $G$  is an undirected graph with atomic edges, the order of vertices representing the edge is arbitrary; thus,  $(v, v') = (v', v)$ . However, in graphs with varying edge lengths or non-atomic edges, the order of vertices may become significant. In such cases, a conflict could arise when agents traverse an edge in opposite directions, but not when they follow each other. Thus, edge directionality should be considered in certain contexts. For our purposes, because our grid roadmap representation consists of atomic edges, we assume that the order of vertices does not matter. In the following sections, we will use vertex conflict notation when discussing conflicts, unless explicitly referring to edge conflicts.

In our grid roadmap representation, we adopt a simplified MRMP conflict checking scheme. Figure 3 illustrates our approach, where we assume a square robot with a width equal to half the edge length of a grid cell in a true grid representation. This square robot size enables us to emulate MAPF behavior within our hybrid representation. Unlike a MAPF grid representation, our hybrid representation considers the robot’s geometry within a roadmap rather than planning across grid cells, as depicted in Figure 2.

For vertex conflict checking, we verify whether any part of the robot’s geometry intersects with obstacles or the occupied space of other robots at each robot’s vertex position in the grid roadmap. For edge conflict checking, we examine the robot’s geometry at intermediate points stationed at the center

of each edge connecting two vertices for intersections with the occupied space of other robots.

Representation resolution significantly influences conflict checking by dictating the level of detail in space discretization. In grid-based methods, a robot typically occupies a single cell, simplifying conflict checks to cell occupancy. However, MRMP introduces complexity with non-uniform roadmaps composed of vertices for robot configurations and edges for configuration sequences, making conflict checking more complex as these edges can contain hundreds of configurations. MRMP's complexity is further increased by considering robot shape and volume, making conflicts more likely not just in the same vertex but also between adjacent vertices as resolution increases.

With our grid roadmap representation, at resolution 1, conflict checking in our grid roadmap resembles MAPF's straightforward grid-based approach. However, at higher resolutions the robot's volume may cover multiple vertices in our grid roadmap representation, and therefore requires thorough checks, similar to that seen in MRMP, for conflict-free paths. Thus, higher representation resolutions allow for a more precise modeling of robot positions while demanding a more intricate conflict checking process.

### B. Objective Function

Two common objective functions used to evaluate MAPF solutions are *makespan* and *sum-of-costs* [25]. *Makespan* is the number of timesteps required for all agents to reach their goal. Given a team plan  $\pi = \{\pi_1, \dots, \pi_n\}$ , the makespan cost of  $\pi$  is:

$$\text{Makespan}(\pi) = \max_{i=1, \dots, n} |\pi_i|$$

*Sum-of-costs*, also known as flowtime, is the sum of timesteps across all agents to reach their goals. The sum-of-costs for a team plan  $\pi$  is defined as:

$$\text{Sum of Costs}(\pi) = \sum_{i=1}^n |\pi_i|$$

For the purpose of this paper, we use sum-of-costs as our objective function for all discussion and experiments.

## III. BACKGROUND

In this section, we will provide a brief overview of various MAPF solvers, introduce the concepts of aggressive and conservative constraints in the context of constraint-based search algorithms, and subsequently delve into a more detailed discussion of Conflict-Based Search (CBS) and Conflict-Based Search with Priorities (CBSw/P).

MAPF has been extensively studied, leading to the development of various types of solvers. Reduction-based solvers transform the MAPF problem into well-known NP-hard problems like Boolean Satisfiability (SAT) and Constraint Satisfaction Problem (CSP) [26]. Examples of reduction-based solvers include MDD-SAT [27], DPLL [28], and MDD-SAT+ID [29]. These solvers utilize methods like integer linear programming or answer set programming to quickly solve the transformed NP-hard problems.

---

### Algorithm 1: Constraint-Based Search Framework

---

**Data:** MAPF Problem Instance

**Result:** Team Plan or Failure

```

1  $CT \leftarrow \emptyset$ 
2 Initialize  $Root$  with low-level path for each agent
3 Insert  $Root$  into  $CT$ ;
4 while  $CT$  not empty
5    $N \leftarrow$  Lowest-cost node in  $CT$ 
6    $X \leftarrow$  Find conflict in  $N$ 
7   if  $N$  has no conflict
8     return  $N.solution$ 
9   for each agent  $a_i$  in  $X$ 
10     $C \leftarrow$  ConflictResolution( $a_i, X$ )
11     $A \leftarrow N$ 
12    Add  $C$  to  $A.constraints$  for  $a_i$ 
13    Update  $A.solution$  with low-level path for  $a_i$ 
14    Update  $A.cost$ 
15    Insert  $A$  into  $CT$ 
16 return Failure

```

---

Rule-based solvers introduce agent-movement rules and include algorithms like TASS [30], push and swap [13], parallel push and swap [31], push and rotate [16], and BIBOX [32]. While rule-based solvers guarantee fast and feasible solutions, they typically return lower-quality solutions.

Search-based algorithms involve A\*-based solvers such as HCA\* and WHCA\* [14], OD and ID [33], EPEA\* [34], M\* [35], and hybrid solvers like ICTS [36], CBS [17], ICBS [37], ECBS [22], CBSw/P and PBS [18], and HC-CBS [19]. Search-based algorithms, although slower compared to other solvers, typically yield higher-quality solutions. However, they face challenges with larger problem sizes due to the exponential state space of MAPF. For a comprehensive survey of other types and classes of MAPF solvers, please refer to [38]–[40].

Lastly, more recently, large neighborhood search (LNS) techniques for MAPF have been used to iteratively refine solutions by selectively destroying and reoptimizing parts of the plan, balancing exploration and exploitation [24]. Variants like LNS2 extend this idea by more sophisticated neighborhood selection and repair strategies, often leading to faster convergence on high-quality solutions in large or complex problem instances [41], [42]. While these methods typically relax completeness and optimality guarantees, their scalability and practical performance make them appealing for real-world applications.

### A. Constraint-Based Search Algorithms

Constraint-based search algorithms are a category of MAPF solvers that aim to find solutions for teams by progressively applying constraints to the state spaces of individual agents. These algorithms navigate the state space, prioritizing states that best optimize a specified objective function. They operate within a common framework, detailed in Algorithm 1.

Constraint-based search algorithms employ both low-level and high-level searches. The low-level search involves a pathfinding algorithm that determines the sequence of actions

**IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.**

for an agent to traverse from its start to goal position, considering a set of constraints. Conflicts are identified in a high-level search, which explores a Conflict Tree (CT). The CT iteratively resolves conflicts by growing sets of constraints. These constraints dictate the valid states during the low-level search in individual agents' state spaces.

The CT is a binary tree. A CT Node, denoted as  $N$ , consists of a solution  $N.solution$ , a cost  $N.cost$ , and a set of constraints  $N.constraints$ . All CT nodes are consistent, meaning the paths they hold always conform to the placed constraints. The high-level search aims to find the lowest-cost node that is valid (consistent and conflict-free).

The algorithm starts by computing independent low-level paths for each agent, creating the initial solution for the root node  $Root$  of the conflict tree  $CT$  (lines 1-3). It then iteratively selects the lowest-cost node  $N$  from the  $CT$  (line 5), identifies conflicts  $X$  in the selected node (line 6), and attempts to resolve these conflicts (line 11). If no conflicts are present, the algorithm returns the solution of the current node. Otherwise, it explores alternative paths for the agents by generating constraints used to generate successor nodes (lines 12-15). The process continues until a solution is found or until the  $CT$  is empty, indicating the absence of a viable solution. For a more in-depth explanation of constraint-based algorithms, please refer to [38].

Constraint-based search algorithms typically employ two main approaches: modifying constraints or adjusting the search process. In some variants, constraint refinement focuses on smarter conflict resolution strategies, such as introducing more refined constraints that prevent future conflicts with minimal impact on the rest of the plan. These modifications emphasize conflict resolution and constraint propagation [17], [18], [43]–[45]. Conflict resolution strategies play a crucial role in guiding the search, ensuring efficient exploration of the state space. The algorithm's effectiveness lies in its ability to balance exploration and exploitation, navigating through the individual state spaces of the agents to find a feasible solution.

Alternatively, search modification algorithms enhance the efficiency of the high-level or low-level search through techniques like heuristic-driven pruning or parallel processing. These methods aim to improve the performance of constraint-based search by incorporating advanced search heuristics and allowing for controlled suboptimality [19], [22], [23], [37], [46]–[50]. Even when algorithms utilize both approaches, their contributions can generally be categorized as either modifications to the constraints or modifications to the search process [51]–[55].

This study focuses on modifications to constraints rather than alterations to the search process. Although constraint-based search algorithms share a common framework, their performance and behavior vary depending on how conflicts are resolved through the application of constraints to the state space. Our approach emphasizes selecting and refining constraints based on key problem features, such as problem size, representation resolution, and representation topology. We do not explore modifications to the search process, such as heuristic-driven improvements, as these are outside the scope of this paper. By tailoring constraints to specific problem

characteristics, we aim to enhance the design and performance of constraint-based search algorithms across diverse domains. We categorize constraints as either aggressive or conservative and, in the following subsections, define these categorizations and discuss the conflict resolution techniques of CBS and CBSw/P.

### B. Aggressive vs. Conservative Constraints

*Conservative constraints* prioritize maintaining a complete search space and are characterized by their localized impact on agent movements. These constraints impose precise, highly specific limitations on agents, aiming to resolve conflicts with minimal disruption to their overall trajectories. They prioritize precision in agent paths by localizing the impact of their constraints. Typically, the precision and limited scope of these constraints result in solutions with more coordinated behavior. However, this approach can lead to longer planning times when numerous constraints are required to reach a solution, as observed in our experiments (Section V-B).

*Aggressive constraints*, on the other hand, prioritize rapid advancement in the search space by constraining a broader range of states, often influencing significant portions of the agent's trajectory. While these constraints may be applied across the entire path, their primary aim is to encompass more states to expedite solution discovery. This emphasis on efficient exploration typically leads to faster planning but may compromise path quality. The broad nature of aggressive constraints typically results in solutions with less coordinated behavior.

Aggressive constraints encourage exploratory behavior, enhancing search breadth, whereas conservative constraints focus on exploitation, efficiently resolving individual conflicts and maintaining solution completeness. Choosing between these strategies requires weighing path quality against planning efficiency within the given context.

We use conservative to mean motion constraints as in vanilla CBS, and aggressive to mean priority constraints as in CBSw/P. These terms describe how conflicts are turned into constraints. They are independent of the high-level search strategy (e.g., best-first, depth-first, weighted, or LNS). Thus, our comparisons isolate the effects of constraint families under a fixed high-level search, rather than conflating them with changes in the search itself.

### C. Conflict-Based Search

In Conflict-Based Search (CBS), when a conflict  $\langle a_i, a_j, v, t \rangle$  is detected between agents  $a_i$  and  $a_j$  at vertex  $v$  during timestep  $t$ , its conflict resolution involves creating two constraints:  $\langle a_i, v, t \rangle$  and  $\langle a_j, v, t \rangle$ . These constraints, termed *motion constraints*, specify that a particular agent is restricted from accessing a vertex or edge at a specific timestep. CBS ensures optimality and a complete search space by generating two constraints that independently limit the low-level searches for agents  $a_i$  and  $a_j$ . Exploring both possibilities guarantees that no states within the search space are overlooked or excluded.

**IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.**

The motion constraints utilized by CBS are classified as conservative constraints. They pinpoint a specific position at a precise moment, minimizing the number of constrained states. This approach ensures the preservation of a complete search space, a hallmark of conservative strategies. Unlike coupled approaches that search joint composite state spaces to maintain completeness, CBS offers distinct advantages by searching the individual state spaces of each agent. Consequently, CBS's state space scales linearly with the number of robots, while its search space expands with the number of conflicts encountered. As a result, CBS's performance is influenced by the problem's complexity, the environment's topology, and the frequency of conflicts during the solving process. Each conflict encountered results in CBS adding two additional child nodes to its CT, leading to increases in the CT size, especially in environments with high-traffic areas.

#### D. Conflict-Based Search with Priorities

In Conflict-Based Search with Priorities (CBSw/P), when a conflict  $\langle a_i, a_j, v, t \rangle$  is detected, its conflict resolution involves creating two *priority constraints*:  $\langle a_i \prec a_j \rangle$  and  $\langle a_j \prec a_i \rangle$ . The first constraint denotes that agent  $a_i$  has a higher priority than agent  $a_j$ . When agent  $a_i$  possesses higher priority than agent  $a_j$ , the path for agent  $a_i$  is planned first. Then, the path for agent  $a_j$  is planned with respect to agent  $a_i$ 's path such that agent  $a_j$  completely avoids agent  $a_i$ 's path. This ensures that after replanning both agents, no conflicts exist between the two agents.

A popular variant of CBSw/P is Priority-Based Search (PBS), which uses the constraint method of CBSw/P alongside a depth-first high-level search. By using the best-first high-level search of CBSw/P, we avoid conflating constraint classifications with search strategies.

Priority constraints, classified as aggressive, require an agent to avoid any intersections with paths of higher-priority agents for their entire journey. Unlike motion constraints that apply to specific movements and timesteps, priority constraints are persistent, preventing an agent from entering states occupied by those with higher priority. These constraints might shift with every planning iteration, reflecting changes in the higher-priority agents' paths. Due to its aggressive constraints, CBSw/P is an incomplete solver that does not maintain a complete search space.

CBSw/P's efficiency is influenced by its Priority Tree (PT) size, which is analogous to CBS's CT but employs priority constraints instead of motion constraints for expansion. The PT is often smaller than the CT due to a bounded number of branches limited by the number of all possible ordered priority pairs. Although the PT has fewer branches, CBSw/P's low-level search typically demands more time due to potentially having to solve multiple pathfinding problems at each node if priority constraints are interconnected, coupled with the higher cost of verifying constraint consistency. CBSw/P is both incomplete and suboptimal; it may fail to find a solution even when one exists and does not guarantee the lowest-cost solution, as it relies on heuristic prioritization rather than exhaustive exploration.

#### E. CBS vs. CBSw/P

While existing research explores the map topologies where CBS and CBSw/P excel, there is a gap in analyzing how their performance is affected by the representation topology and resolution. This absence of analysis poses challenges in extending these findings to the domain of multi-robot motion planning, where representations are typically denser, more intricate, and contain artificial topologies. Our focus on studying CBS and CBSw/P stems from their shared algorithmic structure, utilizing the hierarchical planning approach introduced by CBS, which divides the search into two levels. The primary distinction between these algorithms lies in their conflict resolution and how they constrain agent state spaces. In what follows, we compare motion vs. priority constraints while keeping the high-level search strategy the same (best-first). Analyses that vary the strategy (e.g., DFS, weighted, LNS) are left for future work.

Both CBS and CBSw/P iteratively expand their search spaces by traversing their respective high-level trees. As constraints are imposed, the search space expands accordingly, and the explored states resulting from this expansion are determined by the constraints imposed by the algorithm on their agents.

CBS employs motion constraints to conservatively restrict an agent from accessing a specific configuration at a specific timestep. These constraints promote more coordinated behavior between agents but result in longer planning times due to the increased size of the CT and the incremental nature of constraint application, which restricts the search space one state at a time.

In contrast, CBSw/P utilizes priority constraints, preventing an agent from accessing configurations traversed by higher-priority agents for the entire planning duration. This approach ensures that two agents conflict at most once within a given branch of the priority tree (PT), reducing the conflict-driven search expansion observed in CBS. By aggressively imposing broad path constraints, CBSw/P prioritizes efficient solution discovery over path quality. However, this aggressive constraint application leads to less coordinated behavior between agents.

#### F. Recent Advancements

In this study, we employ the vanilla versions of Conflict-Based Search (CBS) and Priority-Based Search (CBSw/P) to highlight how different constraint mechanisms impact search efficiency. Despite numerous enhancements in the literature, most modern MAPF approaches still build on CBS's motion constraints or CBSw/P's priority constraints. The choice between these paradigms significantly affects scalability, solution quality, and runtime, emphasizing the importance of constraint selection for specific problem domains. However, baseline CBS often struggles with scalability as the number of agents increases, leading to various extensions that modify the search procedure, refine constraints, or balance optimality with runtime. Table I categorizes constraint-based search algorithms based on (1) aggressive (A) vs. conservative (C) constraints, (2) completeness, (3) optimality or bounded suboptimality (B),

## IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

TABLE I  
COMPARISON OF RELATED WORK

## MAPF (Multi-Agent Pathfinding)

Algorithm	Constraint Type	Complete	Optimal	Constraint	Search	Motion	Priority
CBS [17]	C	×	×	×		×	
CBSw/P [18]	A			×			×
PBS [18]	A				×		×
CBS w/ BP [56]	C	×	×		×	×	
CBS w/ Disjoint Splitting [43]	C	×	×	×		~	
CBS w/ PC [37]	C	×	×		×	×	
CBSH [47]	C	×	×		×		
CBSH-RCT [45]	C	×	×	×		~	
CBS-M [57]	C	×	×	×		~	
ECBS [22]	C	×	$\mathcal{B}$		×	×	
EECBS [23]	C	×	$\mathcal{B}$		×	×	
GPBS [58]	A				×		×
HC-CBS [19]	C	×	×		×	×	
IDCBS [46]	C	×	×		×	×	
LaCAM [52]	C	×	×	×	×	×	
Lazy CBS [59]	C	×	×		×	×	
MA-CBS [49]	C	×	×		×	×	

## MAPD (Multi-Agent Pickup and Delivery)

Algorithm	Constraint Type	Complete	Optimal	Constraint	Search	Motion	Priority
LNS-PBS [60]	A	×			×		×
LNS-wPBS [60]	A				×		×

## MAPF-CT (Multi-Agent Pathfinding with Continuous Time)

Algorithm	Constraint Type	Complete	Optimal	Constraint	Search	Motion	Priority
CBICS [61]	C	×	×	×		~	
CCBS [62]	C	×	×	×		~	

## MRMP (Multi-Robot Motion Planning)

Algorithm	Constraint Type	Complete	Optimal	Constraint	Search	Motion	Priority
CBS-MP [63]	C	×	×	×		~	
D-PBS [64]	A			×			×
S2M2 [65]	A				×		×

(4) whether they modify the search or the constraints, and (5) whether they use CBS's motion constraints or CBSw/P's priority constraints. Below, we discuss some key methods in more detail.

CBS is inherently conservative, introducing collision-avoidance constraints only when conflicts occur, minimizing disruption to other agents and timesteps. This localized conflict resolution ensures completeness (it always finds a solution if one exists) and optimality (it guarantees a minimal-cost solution given sufficient time). Several extensions build on this foundation:

- CBS with Disjoint Splitting [43] maintains CBS's conservative approach but reduces branching overhead by introducing positive constraints instead of two negative constraints. This improves CBS's success rate and runtime by up to two orders of magnitude.
- CBS with Heuristics (CBSH) [47] and CBSH-RCT [45] enhance the high-level search using heuristics. CBSH assigns additional costs to conflict tree nodes based on conflict types, improving success rate and runtime by up to 50x. CBSH-RCT further accelerates resolution by applying constraints over extended time windows to target

symmetry conflicts, doubling success rates and improving runtime by an order of magnitude. Both maintain completeness and optimality.

- Meta-Agent CBS (MA-CBS) [49] merges frequently conflicting agents into meta-agents, handling them in a composite state during low-level planning. This approach improves CBS's success rate by a factor of two and enhances runtime efficiency.
- CBS with Bypass (CBS w/ BP) [56] reduces conflict tree expansions by resolving conflicts through alternative, cost-equivalent paths with fewer conflicts, avoiding unnecessary node splitting. This optimization maintains CBS's completeness and optimality guarantees while improving efficiency.
- CBS with Prioritized Conflicts (CBS w/ PC) [37] classifies conflicts as cardinal, semi-cardinal, or non-cardinal based on their impact on solution cost. By prioritizing more problematic conflicts, CBS w/ PC guides the high-level search toward an optimal solution more efficiently while preserving CBS's completeness and optimality.
- Hierarchical Composition CBS (HC-CBS) [19] and its parallel variants improve CBS by resolving conflicts hi-

**IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.**

erarchically, reducing computational complexity through multi-level abstraction. Parallel HC-CBS (PHC-CBS) and Dynamic Parallel HC-CBS (DPHC-CBS) further enhance efficiency by distributing computations across multiple threads to accelerate runtime.

All these methods retain CBS’s conservative constraint style by keeping constraints localized in time and space, ensuring optimal solutions while improving efficiency.

To improve scalability while relaxing strict optimality, several CBS extensions retain the conservative conflict model but introduce modifications that prioritize efficiency over exact solution quality:

- Enhanced CBS (ECBS) [22] and Enhanced Extended CBS (EECBS) [23] introduce a suboptimality bound  $w > 1$ , maintaining CBS’s localized, conflict-triggered constraints while employing aggressive search pruning through inflated heuristics or priority queues. These methods significantly reduce runtime while ensuring solutions remain within a bounded factor of optimal.
- Lazy CBS [59] formulates constraints dynamically rather than predefining all constraints and variables upfront. Instead of branching on every detected conflict, it incrementally adds violated constraints to a partial constraint model, using core-guided search to find feasible minimum-cost plans. This approach improves robustness and runtime, particularly in highly contentious scenarios.

Although these methods improve planning efficiency by pruning or restructuring search strategies, they retain CBS’s conservative constraint formulation. Their primary distinction from vanilla CBS lies in how they explore, expand, and manage the search tree rather than in how they define constraints.

In contrast to CBS, Conflict-Based Search with Priorities (CBSw/P) applies a more aggressive constraint strategy. Agents are assigned a fixed priority ordering, and when a conflict arises, the lower-priority agent must replan its entire trajectory to avoid conflicts with higher-priority agents for the full time horizon. As a result, even a single conflict can lead to broad constraints that reshape an agent’s overall path. Priority-Based Search (PBS) then applies a depth-first high-level search instead of a best-first level search. Several PBS-based extensions build on this approach:

- LNS-PBS and LNS-wPBS [60] integrate large neighborhood search (LNS) with PBS, iteratively modifying and re-optimizing subsets of agent paths. LNS-PBS maintains completeness for well-formed problem instances, focusing on effectiveness and solution quality. LNS-wPBS, however, incorporates a windowed MAPF algorithm that sacrifices completeness guarantees but significantly enhances scalability and stability.
- Greedy PBS (GPBS) [58] employs a greedy strategy to establish or update agent priorities, aiming to minimize collisions while improving efficiency. It introduces techniques such as partial expansions, target reasoning, induced constraints, and soft restarts to further optimize search performance. GPBS demonstrates higher success rates compared to other suboptimal algorithms, partic-

ularly in MAPF instances with small environments or dense obstacles.

These approaches leverage PBS’s aggressive constraint application to improve efficiency, albeit often at the cost of completeness.

Beyond the core CBS and CBSw/P families, several methods extend MAPF to specialized problem variations (see Table I):

- Multi-Agent Pickup and Delivery (MAPD) methods, including certain CBSw/P extensions, introduce dynamic task arrivals and often prioritize responsiveness over completeness or optimality.
- MAPF with Continuous Time (MAPF-CT) generalizes CBS’s local constraints to continuous time intervals rather than discrete timesteps. These methods maintain CBS’s conservative constraint application, preserving completeness and, in many cases, optimality.
- Multi-Robot Motion Planning (MRMP) extends MAPF to higher-dimensional spaces with complex kinematic or dynamic constraints. Many MRMP approaches adapt conflict- or priority-based frameworks to continuous spaces, balancing completeness and computational efficiency.

While CBS and CBSw/P represent the dominant paradigms for motion- vs. priority-based constraints, some methods adopt a more unified constraint-based approach. One such example is Lazy Constraint Addition for MAPF (LaCAM) [52], which encodes all agent movements, collision avoidance, and cost objectives into a single constraint model. Unlike CBS, which branches based on conflicts, or CBSw/P, which assigns global priorities, LaCAM lazily adds constraints only when a partial solution exhibits a collision. This method retains completeness when exhaustively implemented but produces suboptimal solutions. By integrating advanced constraint-solving techniques like clause learning with MAPF-specific collision detection, LaCAM achieves high success rates and low runtimes, particularly in large-agent scenarios.

Each algorithm finds its niche by balancing aggressiveness vs. conservativeness in constraint application, completeness vs. bounded suboptimality, and reliance on explicit high-level/low-level expansions vs. unified constraint models. CBS-based methods remain the gold standard for complete and optimal planning, while CBSw/P-inspired approaches trade completeness for speed and scalability, making them well-suited for large, complex environments.

#### IV. EXPERIMENTAL METHODOLOGY

In the following subsections, we discuss our choices in representation resolution, the identification of representation topology, and our experimental setup. Together, these aspects provide a comprehensive analysis of the behavior of aggressive and conservative constraints under varying problem properties.

##### A. Representation Resolution

The performance of CBS and CBSw/P is significantly influenced by the density and topology of the map. Our grid

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

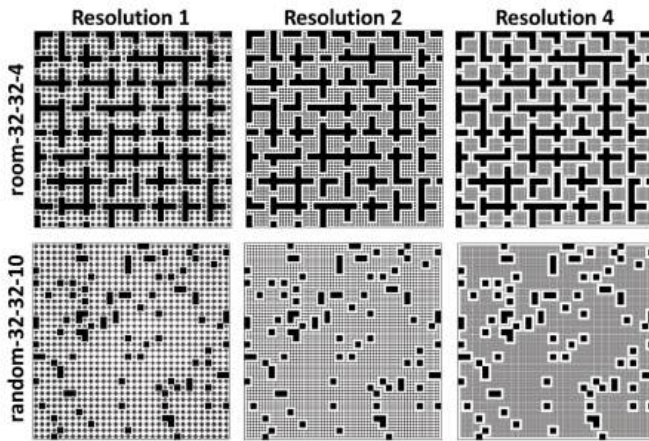


Fig. 4. The grid roadmap representation graph for two sample environments is shown with varying resolutions of 1, 2, and 4.

roadmap-based representation simplifies the inherent complexities of MRMP roadmap representations but also affords the control to enhance the uniform granularity of the representation resolution. The resolutions encompass 1, 2, and 4, where grid roadmap resolution indicates the sampling density within the space while maintaining consistent coverage. Figure 3 provides visual examples of these grid resolutions in a sample environment and Figure 4 provides visual examples of these grid resolutions in some of our benchmark environments. At the lowest resolution level of 1, vertices are placed at the center of cells in true grid representations, reflecting the planning behavior seen in MAPF grid representations. As the resolution increases, the grid roadmap is subdivided to maintain the same coverage area but with a higher density of vertices. Specifically, a grid roadmap at resolution 1 is divided into 2 and 4 subdivisions for resolution levels 2 and 4, respectively.

The representation resolution allows us to achieve finer granularity and detail in the representation and facilitate more precise agent movements while maintaining consistent representation coverage. By adjusting the resolution, we can emulate the behavior observed in MRMP roadmaps with higher edge discretization. For instance, when projecting a path at resolution 1 into higher resolutions, as illustrated in Figure 4, the number of states in the path increases correspondingly. This closely resembles the process of increasing edge discretization in an MRMP roadmap representation. In essence, manipulating the representation resolution offers a flexible means to simulate the characteristics of more complex representations, enabling us to study their effects on algorithm performance and behavior.

This versatility in resolution choices allows for a nuanced approach in addressing MRMP challenges. Our grid roadmap-based resolution exploration allows us to gain insight into the challenges faced in MRMP roadmaps without having to consider the nonuniformity of roadmap edge durations. Our hypothesis suggests that CBS is better suited for roadmap representations with less granularity, particularly in scenarios with fewer agents. Conversely, as representations increase with the number of agents and the resolution increases, CBSw/P is anticipated to outperform CBS. This analysis aims to shed

light on determining the scalability of MAPF solvers into MRMP problems characterized by larger and more complex representations. By understanding the performance dynamics of CBS and CBSw/P in varying resolutions, we can make informed decisions about their applicability in real-world, intricate environments.

### B. Representation Topology

MAPF and MRMP are sensitive to the topology of an environment, influenced by the robot's shape and size relative to environmental features. Narrow passages restrict movement, while open spaces allow more freedom. While practitioners often develop an intuitive understanding of environmental topology through experience, analyzing the topology of abstract representations remains significantly more challenging.

It is important to distinguish between the topology of the environment and the topology of its representation. *Environment topology* refers to the intrinsic geometric and structural properties of the physical space. *Representation topology*, by contrast, refers to the structure encoded in the chosen abstraction—such as a grid, roadmap, or graph—used to approximate the topology of the planning space (e.g., configuration space). This planning space is determined by both the environment and the robot's motion capabilities, including its geometry and degrees of freedom. While many representations aim to preserve the topological structure of the planning space so that the representation topology reflects key features of the environment, methods like sampling-based planning can introduce artifacts that distort or incompletely capture this structure, leading to a mismatch between the representation topology and the environment topology.

MAPF algorithms operate on a fixed representation and do not construct it. As such, when applying MAPF to different planning domains, the relevant topology is not that of the environment itself, but of the representation. This distinction is critical: the same environment can yield multiple representations that emphasize different topological features, as illustrated in Figure 2. Consequently, performance and coordination behavior are shaped more by representation topology than by the physical environment.

In mobile robot settings, the robots typically have low degrees of freedom and simple geometries, allowing structured representations such as grids to align closely with the environment topology, making the relationship between environment topology and representation topology relatively straightforward. In contrast, MRMP for higher-dimensional systems with complex geometries, such as robotic manipulators, typically relies on sampling-based representations, such as Probabilistic Roadmaps (PRM) [66] or Rapidly-exploring Random Trees (RRT) [67]. These methods sample the free configuration space and connect nearby, collision-free configurations. While dense connections may form in open regions and sparse connections in constrained areas, the resulting roadmaps are still often irregular and introduce artificial topological features that obscure the underlying structure. For high-DoF systems, the configuration space topology may have little to no direct correlation with the physical environment topology, further re-

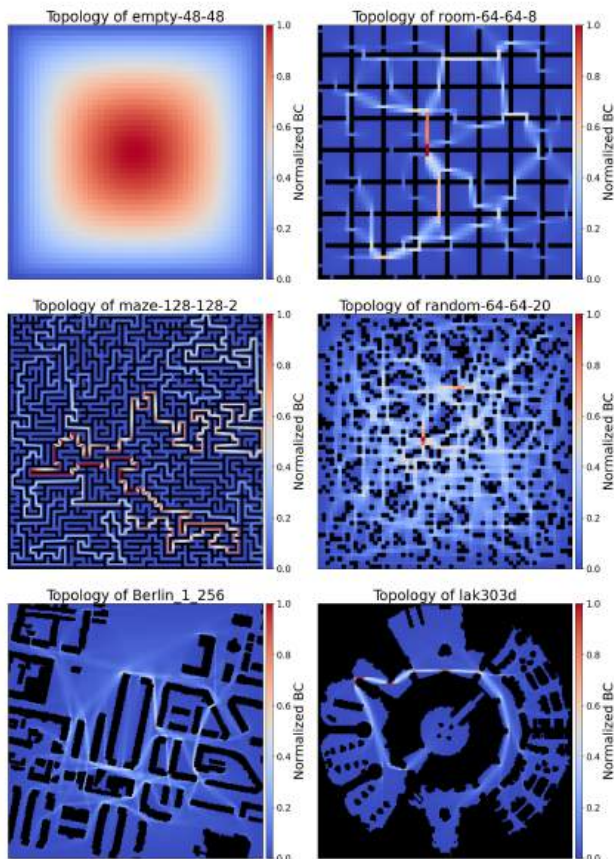


Fig. 5. Betweenness Centrality visualizations for six environments. Blue indicates low betweenness centrality and red indicates high betweenness centrality. Note that the betweenness centrality is not effective for environments devoid of obstacles.

enforcing the need to analyze and understand the representation topology itself.

Classical methods for analyzing environment topology, like Voronoi diagrams and visibility graphs, are less effective for characterizing representation topology. Voronoi diagrams divide space into regions by proximity to obstacles, highlighting open areas and narrow corridors [68]–[70]. Visibility graphs connect obstacle vertices with direct lines of sight, capturing similar features [71], [72]. Both methods require explicit geometric knowledge of the environment and become impractical in high-dimensional configuration spaces, making them unsuitable for determining representation topology.

To address this limitation, betweenness centrality (BC) offers a graph-theoretic approach for inferring topological structure from a given representation [73]. BC quantifies how frequently a node lies on the shortest paths between all pairs of nodes. Mathematically, the BC of a node  $v$  is defined as:

$$BC(v) = \sum_{s \neq v \neq t} \frac{\sigma(s, t | v)}{\sigma(s, t)}$$

where  $\sigma(s, t)$  denotes the number of shortest paths from nodes  $s$  and  $t$ , and  $\sigma(s, t | v)$  is the number of those paths that pass through  $v$ . This measure reflects a node's importance in maintaining connectivity across the network, making it well-suited for identifying critical topological features such as bot-

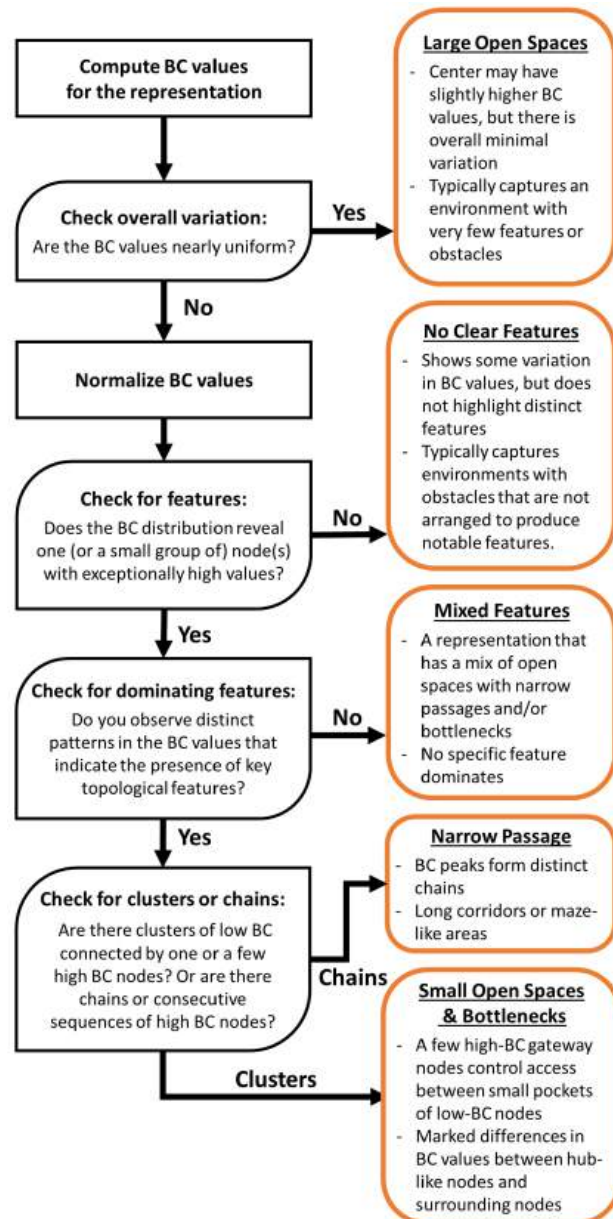


Fig. 6. A flowchart for identifying the dominant topology within a representation using betweenness centrality.

tlenecks and narrow passages. High BC values correspond to frequently traversed nodes, while low values indicate regions with multiple alternate routes.

Figure 5 illustrates the application of BC across several environments used in our experiments. The complete set of visualizations and the code for replicating this analysis are available in our public GitHub Repository<sup>3</sup>. When applied to grid roadmaps, BC highlights narrow passages as chains of high-centrality nodes and identifies bottlenecks as isolated high-centrality nodes surrounded by low-centrality regions.

This method performs well across a variety of environments, with the exception of empty ones. In room and maze environments, BC effectively highlights narrow passages and bottlenecks, identifying key locations likely to experience heavy traversal. In randomly scattered environments, BC identifies a few potential narrow passages but does not reveal distinct

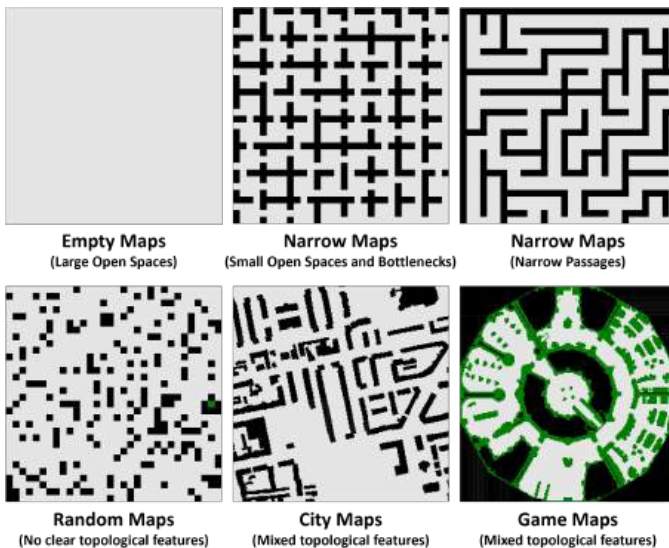


Fig. 7. All maps fall in one of five groups: empty, random, narrow, cities, and games. These groups are characterized by different topological features.

topological features. Small open spaces are marked by clusters of low-centrality nodes, often connected to other open areas via narrow passages or bottlenecks with high-centrality values.

In small open spaces, there tend to be fewer alternative routes for traversing between different points. This results in higher centrality for the central nodes in these small spaces because more paths must pass through them to connect other points. Thus, in these smaller spaces, a few central nodes may control most of the traffic or flow between others. In large open spaces, there are typically more routes between any two points. As a result, the centrality of individual nodes tends to be lower, even for central locations, because traffic can be distributed over many different paths. Consequently, the BC is spread out across many nodes, with fewer nodes acting as critical intermediaries.

In most environments, BC effectively distinguishes between different topological regions. For instance, in the city map (Berlin\_1\_256) and game map (lak303d) shown in Figure 5, BC clearly distinguishes between large open areas and narrow passages. However, in empty environments lacking obstacles, BC tends to highlight the center of the map as the most traversed area, despite the absence of meaningful topological constraints. This limitation arises from the uniformity of the space, which lacks structural features to induce variation in centrality values. Such open environments can be identified by their minimal variance in centrality values prior to normalization.

BC is a simple and robust tool for identifying topological features within graph-based representations. To support its practical use, we provide a flowchart in Figure 6 to guide users in identifying dominant topological patterns. Based on this method, we categorize the environments used in our experiments into five general types according to their topological features.

### C. Experimental Setup

We conduct a comprehensive comparison between CBS and CBSw/P, evaluating their performance in terms of runtime, solution quality (sum-of-costs), and the maximum problem size (number of agents) solved across environments with varying resolutions. Both algorithms are run with a best-first high-level search to ensure that observed differences reflect constraint formulation rather than search strategy. At the low level, we use A\* with a 4-neighbor model and sum-of-costs as the objective. Runtime instance plots show the number of CBS and CBSw/P instances solved within the 15-minute limit, plotted against solution time. Additionally, we record the maximum number of agents solved for each map, enabling a direct comparison of scalability. Our experiments compare CBS vs. CBSw/P under these settings and do not claim generality over all conservative versus aggressive methods or over alternative high-level search strategies.

We leverage benchmarks from [20] and use a total of 27 maps spanning five general categories: empty maps, random maps, narrow maps, real city maps, and video game maps. Figure 7 provides examples of maps from each category while Table II details the individual maps in each group. These maps are grouped based on topological features that we have identified using betweenness centrality. Empty maps contain no obstacles and are characterized by large open spaces. Random maps contain randomly generated obstacles with no clear topological characterization. Narrow maps contain environments crafted to include narrow passages and bottlenecks; these environments are particularly difficult to solve and include room-like and maze-like maps. Real city maps and video game maps contain a combination of open spaces, narrow passages, and bottlenecks. This benchmark set offers comprehensive coverage of diverse topologies, ranging from artificial environments to real-world MAPF applications through real city and video game maps.

For each map, we execute three different resolutions and 25 scenarios. Each scenario consists of pairs of randomly sampled starts and goals. Following the methodology in [20], we begin with a baseline instance of 4 agents. For each subsequent instance, we incrementally add 4 more agents, each with their own start and goal pairs, to the previous instance. This process continues until the solver fails to find a solution for a given instance within the 15-minute runtime limit. We plot the number of solved instances against the time taken. Each algorithm is executed for 25 scenarios, with each scenario tested using resolution 1, resolution 2, and resolution 4 grid roadmaps. This systematic approach allows us to explore the scalability of CBS and CBSw/P across a range of scenarios and grid roadmap resolutions.

Because we keep the high-level search strategy fixed (best-first), our results highlight the effects motion vs. priority effects rather than strategy effects. We do not include depth-first PBS, bounded-suboptimal CBS, or heuristic/learning-augmented variants that are known to shift runtime and success trade-offs. As a result, the crossover points we report may change under the application of those variants.

## IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

TABLE II  
GRAPH AND PROBLEM STATISTICS

	Map (Size)	R	States	CBS -/+		PBS -/+		
Empty	empty- 8-8 (8×8)	1	64	12	24	12	28	
		2	225	12	28	24	32	
		4	841	4	24	20	32	
	empty- 16-16 (16×16)	1	256	12	36	20	40	
		2	961	0	40	32	56	
		4	3721	0	36	24	56	
	empty- 32-32 (32×32)	1	1024	12	68	32	80	
		2	3969	8	68	32	80	
		4	15625	8	40	28	76	
	empty- 48-48 (48×48)	1	2304	16	84	48	84	
		2	9025	16	76	44	100	
		4	35721	12	40	16	60	
Narrow	room- 32-32-4 (32×32)	1	682	8	24	4	28	
		2	1902	4	20	4	28	
		4	5878	0	20	4	24	
	room- 64-64-8 (64×64)	1	3232	8	24	8	28	
		2	11090	0	20	8	24	
		4	40630	0	16	0	20	
	room- 64-64-16 (64×64)	1	3646	4	36	8	32	
		2	13585	4	24	8	24	
		4	52297	0	16	0	16	
	maze- 32-32-2 (32×32)	1	666	4	16	8	20	
		2	1951	0	16	4	16	
		4	6381	0	12	0	12	
	maze- 32-32-4 (32×32)	1	790	4	16	4	16	
		2	2695	0	16	5	16	
		4	9853	0	12	4	12	
	maze- 128-128-2 (128×128)	1	10858	0	12	0	12	
		2	32383	0	4	0	4	
		4	107437	0	0	0	0	
	maze- 128-128-10 (128×128)	1	14818	0	20	0	20	
		2	56143	0	16	0	16	
		4	218317	0	4	0	4	
	Cities	Berlin- 1-256 (256×256)	1	47536	4	88	24	92
			2	182171	4	60	8	48
			4	712615	0	36	0	36
Boston- 0-256 (256×256)		1	47747	12	64	8	52	
		2	181232	4	40	4	32	
		4	705218	0	12	0	12	
Paris- 1-256 (256×256)		1	47216	0	68	24	64	
		2	179271	0	44	4	44	
		4	697685	0	24	0	24	
Random		random- 32-32-10 (32×32)	1	922	12	44	32	60
			2	3185	4	44	28	60
			4	11575	4	40	28	60
	random- 32-32-20 (32×32)	1	819	12	40	24	44	
		2	2468	8	40	24	44	
		4	8040	4	28	20	40	
	random- 64-64-10 (64×64)	1	3687	20	84	44	88	
		2	12830	8	68	28	88	
		4	46764	4	56	12	60	
	random- 64-64-20 (64×64)	1	3270	8	52	20	76	
		2	10031	8	48	20	48	
		4	33225	8	36	20	48	
	Games	ht-chantry (141×162)	1	7461	12	28	12	28
			2	27912	4	20	4	20
			4	107742	0	16	0	16
		ht-mans ion-n (270×133)	1	8959	4	40	12	40
			2	33376	4	32	0	24
			4	128554	0	20	0	16
		lak303d (194×194)	1	14784	4	24	4	24
			2	54938	0	16	0	16
			4	211230	0	8	0	8
		lt-gallows templar (180×251)	1	10021	8	36	12	36
			2	37637	8	36	8	32
			4	145549	0	16	0	20
den312d (81×65)		1	2445	4	28	16	32	
		2	8779	4	28	8	28	
		4	33105	0	20	4	24	
ost003d (194×194)		1	13214	4	32	4	32	
		2	49986	0	20	0	20	
		4	194168	0	12	0	12	
brc202d (481×530)		1	43151	0	16	0	16	
		2	162968	0	8	0	8	
		4	632432	0	4	0	4	
den520d (257×256)		1	28178	0	48	4	28	
		2	108918	0	24	0	24	
		4	427970	0	16	0	16	
w-wound edcoast (578x642)	1	34002	4	20	4	16		
	2	127838	0	8	0	8		
	4	495024	0	0	0	0		

## V. RESULTS AND DISCUSSION

In this section, we discuss how to interpret our plots and synthesize our findings into key takeaways to provide a concise summary of our study. We recommend reading through this section while viewing the flowchart shown in Figure 1.

## A. Plot and Table Analysis

We compare motion constraints (CBS) and priority constraints (CBSw/P) while holding the high-level search strategy fixed (best-first) so that our findings reflect differences in constraint formulation. Our results are presented in Table II and the plots in Figures 8 to 12. Table II provides insights

into the size of the environments, the number of states within the state space, the resolution ( $R$ ), and the minimum to maximum problem sizes ( $-/+$ ) successfully solved by both CBS and CBSw/P. The main takeaway from this table should be the increase in states as resolution increases and the maps in each grouping. The plots in Figures 8 to 12 offer an average perspective based on map groupings and present runtime instances, average success rates, and sum-of-costs ratios, facilitating a comparison between CBSw/P and CBS. For a detailed exploration of the same statistics on an individual map basis, along with raw data, please refer to our public GitHub repository<sup>3</sup>.

These plots are averaged over map groupings and scenarios.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

Scenarios exhibit varying difficulty due to random generation, which allows us to draw conclusions independent of problem difficulty. Some scenarios are easier due to shorter or non-conflicting paths, while others pose increased difficulty with longer and conflicting paths. Map groupings are based on topological features, with each grouping containing maps of different sizes. By grouping maps according to topology, we can make conclusions that are independent of environment size. These averaged grouped plots effectively highlight key trends relevant to our analysis on the influence of environment representation topology and representation resolution, while minimizing the impact of environment size and problem difficulty variations.

For our runtime analysis, we present runtime instance plots, which show the number of instances solved within the 15-minute limit across all maps and scenarios in a given group. In these experiments, the number of robots was increased in increments of four agents, so each successive solved instance represents a configuration with four more robots than the previous one. These plots avoid the skew introduced by averaging runtimes, where varying scenario difficulty can produce irregular peaks and dips as problem sizes increase and success rates decline. By counting solved instances directly, runtime instance plots provide a clearer view of algorithm performance and scalability. The x-axis of these plots is logarithmic.

We calculate the success rate by dividing the total number of solved scenarios by the total number of possible scenarios across all maps in a given group. Lastly, for our cost ratio plots, we compute the cost ratio between CBSw/P and CBS for scenarios where both algorithms successfully solved the problem. If either algorithm fails to solve a scenario, the cost ratio is not reported, as it cannot be calculated.

### B. Analysis of Results

In this subsection, we consolidate our findings by examining the plots corresponding to our map groupings and elucidating the decision-making process outlined in our flowchart (Figure 1). Our discussion summarizes vanilla CBS and CBSw/P under our evaluation settings and should be viewed as a first-pass heuristic rather than a universal prescription for all conservative and aggressive methods.

1) *Large Open Space Environments*: Figure 8 displays the runtime instance, average success rate, and cost ratios for the empty group of maps, characterized by representations containing vast open spaces devoid of interesting features. This group poses relatively easier challenges compared to other map groupings due to the absence of obstructive elements.

Our analysis consistently shows that CBSw/P achieves higher success rates and comparable or quicker runtimes than CBS across different problem sizes and resolutions. This superior performance comes with a slight compromise in solution quality, as indicated by a maximum suboptimal cost of only about 1.015 times the optimal. In representations with large open spaces, CBSw/P matches or exceeds CBS in both success rate and runtime, irrespective of the problem size or resolution. Therefore, in such settings, we suggest utilizing aggressive constraints for both MAPF and MRMP tasks.

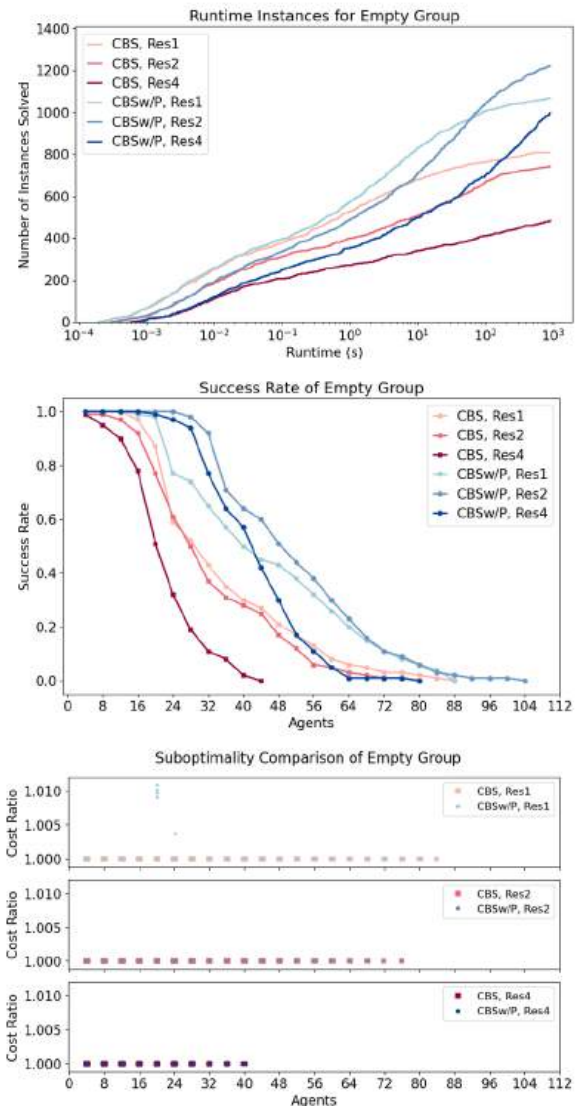


Fig. 8. Analysis of runtime instances (with varying x-axes), success rate, and cost ratios for the Empty Group. The runtime instance plots display instances ranging from 4 to 104 agents.

The aggressive nature of CBSw/P’s constraints aids in the efficient discovery of team solutions, particularly with increasing numbers of agents. Moreover, as resolution increases, the benefits of using aggressive constraints become more pronounced. Thus, regardless of the problem’s scale or representation resolution, we recommend using aggressive constraints because they offer superior scalability without any significant degradation in solution quality. Notably, CBSw/P achieves better scalability with a resolution 2 grid roadmap compared to a resolution 1 grid roadmap. By increasing the number of states in the representation, agents can make smaller movements, which is particularly beneficial in scenarios with higher agent densities. While the resolution 2 grid roadmap results in higher runtimes due to the increased number of possible states, it also offers improved scalability with respect to problem size.

The advantage of aggressive constraints in open spaces lies

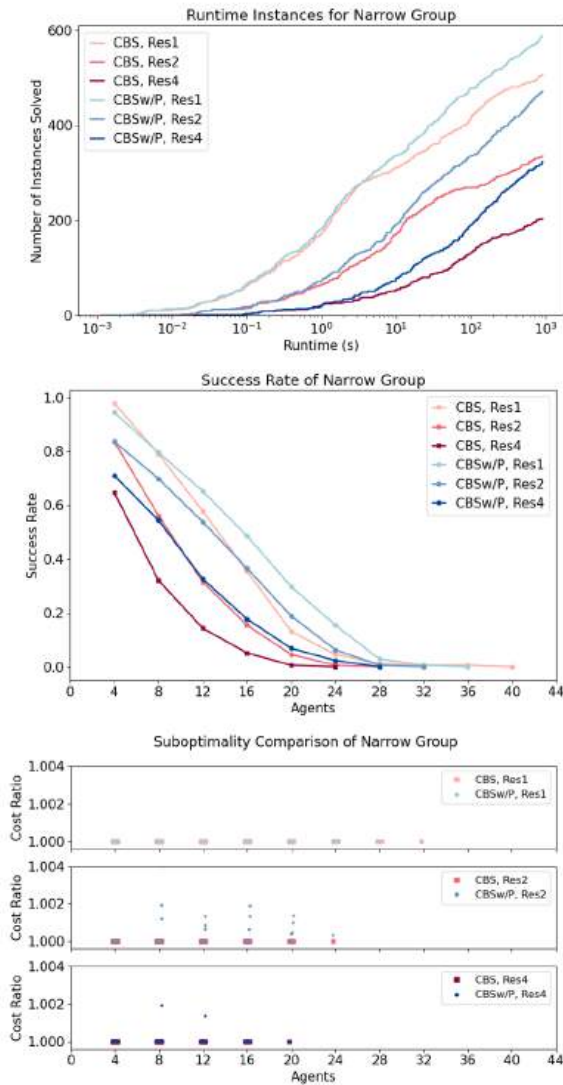


Fig. 9. Analysis of runtime instances (with varying x-axes), success rate, and cost ratios for the Narrow Group. The runtime instance plots display instances ranging from 4 to 40 agents.

in their emphasis on rapid exploration of the search space. They navigate these areas efficiently without being impeded by numerous conflicts, unlike conservative constraints, which may spend excessive time analyzing unnecessary conflicts. Aggressive constraints perform particularly well in scenarios where high levels of coordination are not required. As a result, their inherent characteristics align naturally with open-space representations, making them a more suitable and efficient choice for solving problems in such contexts compared to conservative constraints.

2) *Narrow Environments*: Figure 9 presents the results for the narrow group of maps, which are characterized by numerous narrow passages or bottlenecks coupled with small open spaces. These environments have historically presented formidable challenges due to obstacle configurations that obstruct the traversal of multiple robots, significantly complicating coordination efforts.

In navigating representations with narrow passages, our

analysis begins by considering problem size and representation resolution. For scenarios involving a small number of agents, conservative constraints prove advantageous in ensuring more coordinated behavior between agents. Particularly in cases where both the number of agents is small and the representation resolution is low, conservative constraints demonstrate comparable runtimes and success rates to aggressive constraints.

However, as the number of agents navigating congested regions increases, conservative constraints struggle to find solutions efficiently due to their meticulous constraint application, leading to longer runtimes. In contrast, the scalability of aggressive constraints becomes increasingly evident as problem size grows, making them a more practical choice for larger-scale scenarios, regardless of representation resolution, runtime considerations, or completeness trade-offs.

When evaluating representation resolution alongside a relatively small number of agents, the choice between constraint types depends on its impact on planning time. In MRMP instances with high-resolution representations, aggressive constraints can increase the likelihood of finding a solution by reducing runtime, which may help avoid resource exhaustion. However, aggressive constraints are often incomplete and may not always guarantee a valid solution. In contrast, with lower representation resolutions, conservative constraints provide the advantage of ensuring paths with more coordinated behavior while maintaining runtimes comparable to aggressive constraints. However, we recommend choosing between the two based on the desired completeness and planning time, depending on the specific needs of your problem.

This analysis underscores the intricate relationship between problem attributes, solution strategies, and constraint selection. While conservative constraints excel in ensuring precision, completeness, and optimality, particularly in smaller-scale scenarios, the scalability and exploratory nature of aggressive constraints make them better suited for complex problem instances characterized by numerous agents. Aligning constraint selection with problem attributes enables practitioners to navigate multi-agent systems effectively, optimizing efficiency across diverse representations.

3) *Random Environments*: Random environments are characterized by the arbitrary distribution of obstacles, lacking any predominant topological feature. Figure 10 showcases the results derived from our analysis of these random environments. The absence of specific topological features in these representations poses challenges in predicting optimal strategies.

Aggressive constraints offer a practical approach for navigating random representations by swiftly eliminating similar paths, thereby improving computational efficiency. This mirrors their effectiveness in open-space representations, where scalability is often a priority. Since representations capturing random environments typically require low levels of coordination, aggressive constraints excel in these scenarios. However, this comes at the cost of completeness. When guaranteeing a valid solution is a priority, conservative constraints provide an alternative. However, applying conservative constraints in representations with inherently low coordination requirements

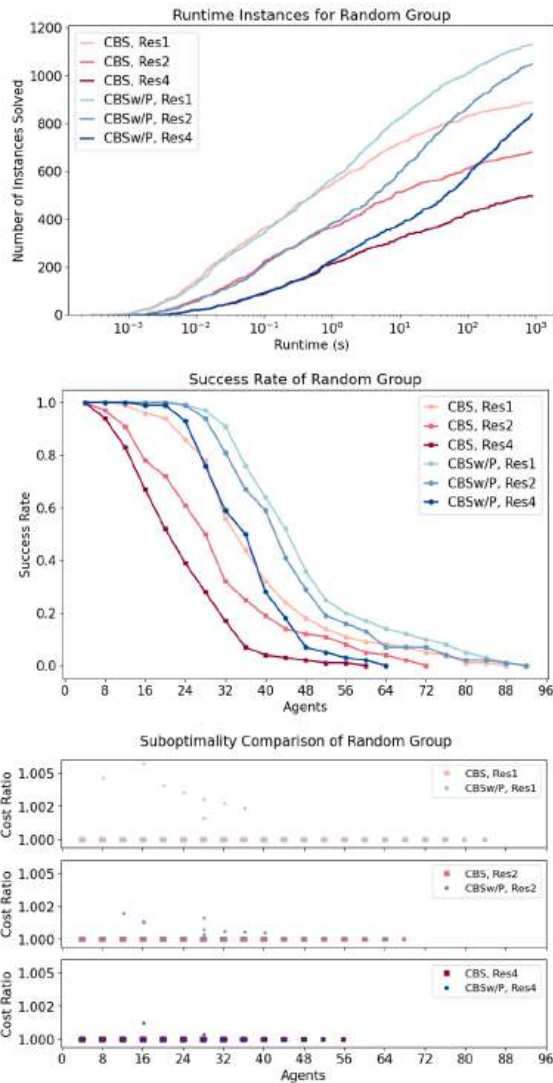


Fig. 10. Analysis of runtime instances (with varying x-axes), success rate, and cost ratios for the Random Group. The runtime instance plots display instances ranging from 4 to 92 agents.

generally increases computational demands, limiting scalability. As a result, aggressive constraints remain the preferable choice in most cases, as they significantly enhance the likelihood of finding viable solutions within a reasonable timeframe.

In the broader context of MAPF and MRMP, aggressive constraints are generally the optimal choice for representations lacking distinguishing features, as they strike a balance between computational efficiency and solution feasibility. For users prioritizing completeness, particularly in smaller problem instances or when working with low-resolution roadmaps, conservative constraints can be a viable option. Ultimately, the choice of constraints should align with the user's specific performance goals and the demands of the task at hand.

4) *Mixed Environments*: Mixed environments encompass various topological features such as open spaces, narrow passages, and bottlenecks. The cities and games groupings exemplify representations that amalgamate these diverse characteristics and their results are shown in Figures 11 and 12,

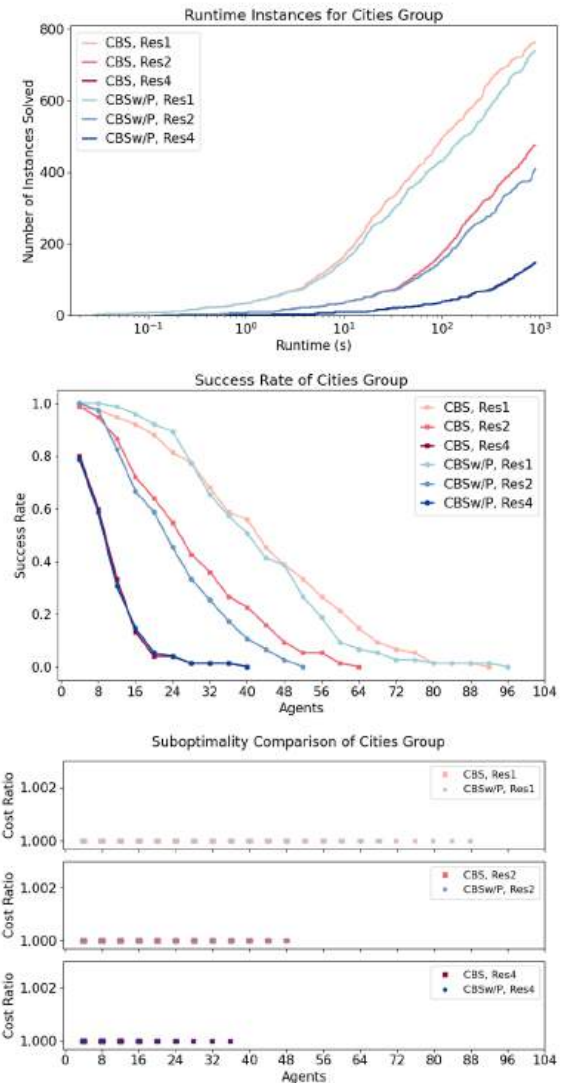


Fig. 11. Analysis of runtime instances (with varying x-axes), success rate, and cost ratios for the Cities Group. The runtime instance plots display instances ranging from 4 to 96 agents

respectively.

In terms of runtime and success rates, conservative constraints demonstrate comparable or superior performance to aggressive constraints as problem size and representation resolution increase. Conservative constraints provide slightly higher success rates with lower planning times, regardless of problem size or resolution. Therefore, in representations with diverse topological features, we recommend using conservative constraints over aggressive ones.

Conservative constraints perform well in such representations by efficiently identifying and resolving conflicts with a few targeted constraints. Their ability to enforce more coordinated behavior with a few targeted constraints allows conflicts to be addressed quickly, making them particularly advantageous in representations where different regions require varying levels of precision or where conflict paths demand more nuanced conflict resolution. Given their consistent performance, often matching or exceeding that of aggressive

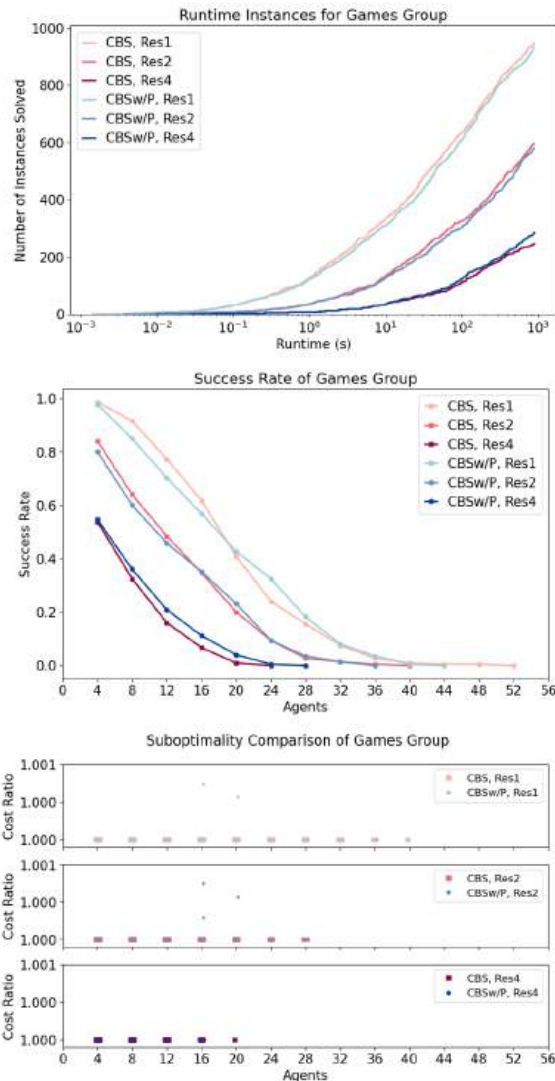


Fig. 12. Analysis of runtime instances (with varying x-axes), success rate, and cost ratios for the Games Group. The runtime instance plots display instances ranging from 4 to 52 agents.

constraints, conservative constraints are the preferable choice for mixed topological representations.

While aggressive constraints can achieve similar results, they tend to incur higher runtimes. As the number of agents increases, the placement of numerous priority constraints to resolve localized conflicts can lead to excessive iterations of planning and validation. Although priority constraints remain a viable approach, the completeness guarantees of conservative constraints make them a more reliable and effective option in comparison.

### C. Key Takeaways

Our study highlights the critical role of coordination requirements in selecting appropriate constraints for constraint-based search. When representations impose high coordination demands, conservative constraints are often preferable, as they methodically resolve conflicts and preserve completeness. Conversely, in scenarios with lower coordination requirements,

aggressive constraints provide a more efficient alternative by rapidly eliminating conflicts and reducing planning overhead.

Our comparisons fix the high-level search to best-first in order to isolate the effects of constraint formulation; other strategies (e.g., depth-first PBS, bounded suboptimal variants, etc.) can shift success rates and scalability. The heuristics and guidelines we provide are intended as a first-pass reference for practitioners when selecting among constraint-based methods. Another thing to note is that our betweenness-centrality topology measure is informative on many maps but less reliable in empty environments, so trends on empty maps should be interpreted with caution. Finally, while the grid roadmap discretization reduces representation artifacts, it is still an abstraction of continuous MRMP spaces. With these considerations in mind, we now summarize our findings from comparing vanilla CBS with CBSw/P.

In mixed representations containing localized regions of dense agent interaction, conservative constraints excel by focusing conflict resolution precisely where it is needed. In contrast, open environments or those lacking distinct topological features benefit from aggressive constraints, which expedite solution discovery by minimizing coordination requirements and pruning large portions of the search space.

However, in representations with narrow passages or bottlenecks, further considerations arise. Although conservative constraints offer more coordinated behavior, they can lead to exponential growth in the constraint tree. This occurs because conservative constraints can repeatedly reintroduce and resolve conflicts between the same agent pairs across different branches during replanning iterations. In contrast, aggressive constraints avoid this redundancy by ensuring that each agent pair conflicts at most once within any given branch. This property allows aggressive constraints to limit the branching factor and reduce search complexity, especially in highly coupled environments or as team size increases. Nevertheless, due to their incompleteness, aggressive constraints may struggle in topologically constrained settings, where more conservative conflict resolution is necessary to guarantee solution validity.

The insights from our previous subsection are summarized into an easily digestible decision flowchart, shown in Figure 1. In this subsection, we will generally follow this flowchart to have a high-level discussion of the results and summarize key takeaways.

1) *Representation Topology*: Our observations indicate that representations with large open spaces or a lack of distinct features benefit from aggressive constraints due to their scalability and runtime advantages. In contrast, for representations with a mix of features, conservative constraints are recommended, as they guarantee high-quality solutions while offering comparable or superior scalability and runtime performance. For environment representations dominated by narrow passages or small open spaces with bottlenecks, both the problem size and representation resolution should guide the choice between aggressive and conservative constraints.

2) *Scale and Granularity Factors*: For settings with large problem sizes or high representation resolutions, aggressive constraints are preferable to conservative constraints due to their superior scalability and computational efficiency. By

**IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.**

imposing fewer constraints, they can identify valid paths more quickly, albeit at the cost of completeness. Nevertheless, they remain effective for solving large-scale problems in dense representations with limited computational resources. In contrast, employing conservative constraints in such contexts is less advisable, as the need to specify exact states can result in an excessive number of constraints, increasing computational overhead and delaying solution discovery. For scenarios with smaller problem sizes and lower representation resolutions, the choice of constraints should be guided by the planner's desired balance between runtime efficiency and solution completeness.

Problem size can be determined by agent density, which refers to the concentration of agents within the traversable free space of an environment. This measure quantifies how crowded an environment is with agents. As a general guideline, "many agents" typically refers to problems with an agent density greater than 10–20% of the traversable free space. However, since constraint-based search algorithms scale based on the number of conflicts encountered, a rough estimate of whether a problem has many or few agents is sufficient for the purposes of this flowchart.

3) *Execution Performance Metrics*: Lastly, the choice of constraints should be guided by the planner's desired runtime performance and completeness requirements. Solution quality should not be a primary factor in this decision, as the differences in solution optimality between constraint types are negligible. Instead, the focus should be on balancing computational efficiency with solution guarantees. If computational resources are limited or completeness is not a requirement, aggressive constraints are preferable due to their faster planning times and scalability. Conversely, if guaranteeing a valid solution is critical and longer planning times are acceptable, conservative constraints should be used to ensure completeness at the cost of increased computational overhead.

## VI. CONCLUSION

This paper investigates how constraint formulation influences the behavior of constraint-based search for MAPF and MRMP. We compared vanilla CBS and CBSw/P under a fixed best-first high-level search. Our results show that environments with high coordination demands tend to favor conservative constraints, which systematically resolve conflicts and preserve completeness, whereas environments with lower coordination demands often benefit from aggressive constraints, which reduce repeated conflict expansion and planning overhead.

We synthesize these findings into the decision flowchart in Figure 1, offering a practical first-pass guide for method selection in MAPF and MRMP applications. These conclusions are limited to the methods and settings studied and should not be interpreted as universal prescriptions for all conservative versus aggressive methods. Alternative high-level strategies and modern enhancements may shift success rates and scalability. Future work could build on this preliminary study by factorially separating constraint formulation from search strategy, incorporating state-of-the-art variants, and broadening topology characterization beyond betweenness centrality and grid-roadmap abstractions.

Looking ahead, advancements in constraint-based search should account for representation features and application-specific requirements. Designing universally flexible MAPF algorithms remains an appealing but difficult challenge. Effective algorithm development must weigh solver design against target environments, problem scales, resolution needs, and the trade-offs between solution quality and efficiency. Although this study did not address noise or uncertainty, we expect the presented insights to extend to such scenarios, which warrant further investigation. By framing constraints as conservative or aggressive, developers can make more informed design choices, ultimately broadening the effectiveness and applicability of MAPF algorithms.

## ACKNOWLEDGMENT

This work was supported in part by the U.S. National Science Foundation's "Expeditions: Mind in Vitro: Computing with Living Neurons" under award No. IIS-2123781, and by the IBM-Illinois Discovery Accelerator Institute and the Center for Networked Intelligent Components and Environments (C-NICE) at the University of Illinois.

The work of Hannah Lee was supported in part by the National Science Foundation Graduate Research Fellowship under Grant No. 2020297899. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Marco Morales is supported in part by the Asociación Mexicana de Cultura A.C.

## REFERENCES

- [1] D. Halperin, J.-C. Latombe, and R. H. Wilson, "A general framework for assembly planning: The motion space approach," in *Proceedings of the fourteenth annual symposium on Computational geometry*, pp. 9–18, 1998.
- [2] B. O. Nnaji, *Theory of automatic robot assembly and programming*. Springer Science & Business Media, 1993.
- [3] K. Brown, O. Peltzer, M. A. Sehr, M. Schwager, and M. J. Kochenderfer, "Optimal sequential task assignment and path finding for multi-agent robotic assembly planning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 441–447, IEEE, 2020.
- [4] S. Rodriguez and N. M. Amato, "Behavior-based evacuation planning," in *2010 IEEE International Conference on Robotics and Automation*, pp. 350–355, IEEE, 2010.
- [5] T. Balch and R. C. Arkin, "Behavior-based formation control for multi-robot teams," *IEEE transactions on robotics and automation*, vol. 14, no. 6, pp. 926–939, 1998.
- [6] H. G. Tanner, G. J. Pappas, and V. Kumar, "Leader-to-formation stability," *IEEE Transactions on robotics and automation*, vol. 20, no. 3, pp. 443–455, 2004.
- [7] S. Kloder and S. Hutchinson, "Path planning for permutation-invariant multirobot formations," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 650–665, 2006.
- [8] J. Li, K. Sun, H. Ma, A. Felner, T. Kumar, and S. Koenig, "Moving agents in formation in congested environments," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 11, pp. 131–132, 2020.
- [9] S. Liu, L. Wen, J. Cui, X. Yang, J. Cao, and Y. Liu, "Moving forward in formation: A decentralized hierarchical learning approach to multi-agent moving together," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4777–4784, IEEE, 2021.
- [10] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous robots*, vol. 8, pp. 325–344, 2000.

## IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

- [11] D. Rus, B. Donald, and J. Jennings, "Moving furniture with teams of autonomous robots," in *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, vol. 1, pp. 235–242, IEEE, 1995.
- [12] J. Yu and S. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 27, pp. 1443–1449, 2013.
- [13] R. Luna and K. E. Bekris, "Push and swap: Fast cooperative path-finding with completeness guarantees," in *IJCAI*, pp. 294–300, 2011.
- [14] D. Silver, "Cooperative pathfinding," in *Proceedings of the aai conference on artificial intelligence and interactive digital entertainment*, vol. 1, pp. 117–122, 2005.
- [15] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *2011 IEEE international conference on robotics and automation*, pp. 5628–5635, IEEE, 2011.
- [16] B. De Wilde, A. W. Ter Mors, and C. Witteveen, "Push and rotate: a complete multi-agent pathfinding algorithm," *Journal of Artificial Intelligence Research*, vol. 51, pp. 443–492, 2014.
- [17] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.
- [18] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 7643–7650, 2019.
- [19] H. Lee, J. Motes, M. Morales, and N. M. Amato, "Parallel hierarchical composition conflict-based search for optimal multi-agent pathfinding," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7001–7008, 2021.
- [20] 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 *Proceedings of the International Symposium on Combinatorial Search*, vol. 10, pp. 151–158, 2019.
- [21] B. J. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for manipulation with motion primitives," in *2010 IEEE international conference on robotics and automation*, pp. 2902–2908, IEEE, 2010.
- [22] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 5, pp. 19–27, 2014.
- [23] J. Li, W. Ruml, and S. Koenig, "Eecbs: A bounded-suboptimal search for multi-agent path finding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, pp. 12353–12362, 2021.
- [24] J. Li, Z. Chen, D. Harabor, P. J. Stuckey, and S. Koenig, "Anytime multi-agent path finding via large neighborhood search," in *International Joint Conference on Artificial Intelligence 2021*, pp. 4127–4135, Association for the Advancement of Artificial Intelligence (AAAI), 2021.
- [25] P. Surynek, A. Felner, R. Stern, and E. Boyarski, "An empirical comparison of the hardness of multi-agent path finding under the makespan and the sum of costs objectives," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 7, pp. 145–146, 2016.
- [26] R. Barták and J. Švancara, "On sat-based approaches for multi-agent path finding with the sum-of-costs objective," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 10, pp. 10–17, 2019.
- [27] P. Surynek, A. Felner, R. Stern, and E. Boyarski, "Efficient sat approach to multi-agent path finding under the sum of costs objective," in *Proceedings of the twenty-second european conference on artificial intelligence*, pp. 810–818, 2016.
- [28] M. Čapek and P. Surynek, "Dpll (mapf): an integration of multi-agent path finding and sat solving technologies," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 12, pp. 153–155, 2021.
- [29] P. Surynek, J. Švancara, A. Felner, and E. Boyarski, "Integration of independence detection into sat-based optimal multi-agent path finding—a novel sat-based optimal mapf solver," in *International Conference on Agents and Artificial Intelligence*, vol. 2, pp. 85–95, SciTePress, 2017.
- [30] M. Khorshid, R. Holte, and N. Sturtevant, "A polynomial-time algorithm for non-optimal multi-agent pathfinding," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 2, pp. 76–83, 2011.
- [31] Q. Sajid, R. Luna, and K. Bekris, "Multi-agent pathfinding with simultaneous execution of single-agent primitives," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 3, pp. 88–96, 2012.
- [32] P. Surynek, "A novel approach to path planning for multiple robots in bi-connected graphs," in *2009 IEEE International Conference on Robotics and Automation*, pp. 3613–3619, IEEE, 2009.
- [33] T. Standley, "Finding optimal solutions to cooperative pathfinding problems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, pp. 173–178, 2010.
- [34] A. Felner, M. Goldenberg, G. Sharon, R. Stern, T. Beja, N. Sturtevant, J. Schaeffer, and R. Holte, "Partial-expansion a\* with selective node generation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, pp. 471–477, 2012.
- [35] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial intelligence*, vol. 219, pp. 1–24, 2015.
- [36] G. Sharon, R. Stern, M. Goldenberg, and A. Felner, "The increasing cost tree search for optimal multi-agent pathfinding," *Artificial intelligence*, vol. 195, pp. 470–495, 2013.
- [37] E. Boyarski, A. Felner, R. Stern, G. Sharon, E. Shimony, O. Bezalel, and D. Tolpin, "Improved conflict-based search for optimal multi-agent path finding," in *24th International Joint Conference on Artificial Intelligence, IJCAI 2015*, 2015.
- [38] A. Felner, R. Stern, S. Shimony, E. Boyarski, M. Goldenberg, G. Sharon, N. Sturtevant, G. Wagner, and P. Surynek, "Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 8, pp. 29–37, 2017.
- [39] R. Stern, "Multi-agent path finding—an overview," *Artificial Intelligence: 5th RAAI Summer School, Dolgoprudny, Russia, July 4–7, 2019, Tutorial Lectures*, pp. 96–115, 2019.
- [40] E. Lejeune, S. Sarkar, and L. Jezequel, "A survey of the multi-agent pathfinding problem," tech. rep., Technical Report, Accessed July, 2021.
- [41] Y. Wang, T. Duhan, J. Li, and G. Sartoretti, "Lns2+ rl: Combining multi-agent reinforcement learning with large neighborhood search in multi-agent path finding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, pp. 23343–23350, 2025.
- [42] J. Li, Z. Chen, D. Harabor, P. J. Stuckey, and S. Koenig, "Mapf-Ins2: Fast repairing for multi-agent path finding via large neighborhood search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 10256–10265, 2022.
- [43] J. Li, D. Harabor, P. J. Stuckey, A. Felner, H. Ma, and S. Koenig, "Disjoint splitting for multi-agent path finding with conflict-based search," in *Proceedings of the international conference on automated planning and scheduling*, vol. 29, pp. 279–283, 2019.
- [44] J. Li, D. Harabor, P. J. Stuckey, H. Ma, and S. Koenig, "Symmetry-breaking constraints for grid-based multi-agent path finding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 6087–6095, 2019.
- [45] J. Li, G. Gange, D. Harabor, P. J. Stuckey, H. Ma, and S. Koenig, "New techniques for pairwise symmetry breaking in multi-agent path finding," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, pp. 193–201, 2020.
- [46] E. Boyarski, A. Felner, D. Harabor, P. J. Stuckey, L. Cohen, J. Li, and S. Koenig, "Iterative-deepening conflict-based search," in *International Joint Conference on Artificial Intelligence-Pacific Rim International Conference on Artificial Intelligence 2020*, pp. 4084–4090, Association for the Advancement of Artificial Intelligence (AAAI), 2020.
- [47] A. Felner, J. Li, E. Boyarski, H. Ma, L. Cohen, T. S. Kumar, and S. Koenig, "Adding heuristics to conflict-based search for multi-agent path finding," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 28, pp. 83–87, 2018.
- [48] J. Li, A. Felner, E. Boyarski, H. Ma, and S. Koenig, "Improved heuristics for multi-agent path finding with conflict-based search.," in *IJCAI*, vol. 2019, pp. 442–449, 2019.
- [49] G. Sharon, R. Stern, A. Felner, and N. Sturtevant, "Meta-agent conflict-based search for optimal multi-agent path finding," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 3, pp. 97–104, 2012.
- [50] J. Švancara, M. Vlk, R. Stern, D. Atzmon, and R. Barták, "Online multi-agent pathfinding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 7732–7739, 2019.
- [51] J. Li, P. Surynek, A. Felner, H. Ma, T. S. Kumar, and S. Koenig, "Multi-agent path finding for large agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 7627–7634, 2019.
- [52] K. Okumura, "Lacam: Search-based algorithm for quick multi-agent pathfinding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 11655–11662, 2023.
- [53] A. Andreychuk, K. Yakovlev, P. Surynek, D. Atzmon, and R. Stern, "Multi-agent pathfinding with continuous time," *Artificial Intelligence*, vol. 305, p. 103662, 2022.

## IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

- [54] J. Jin, Y. Zhang, Z. Zhou, M. Jin, X. Yang, and F. Hu, "Conflict-based search with d\* lite algorithm for robot path planning in unknown dynamic environments," *Computers and Electrical Engineering*, vol. 105, p. 108473, 2023.
- [55] A. Andreychuk, "Multi-agent path finding with kinematic constraints via conflict based search," in *Artificial Intelligence: 18th Russian Conference, RCAI 2020, Moscow, Russia, October 10–16, 2020, Proceedings 18*, pp. 29–45, Springer, 2020.
- [56] E. Boyarski, A. Felner, G. Sharon, and R. Stern, "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*, vol. 25, pp. 47–51, 2015.
- [57] H. Zhang, J. Li, P. Surynek, T. S. Kumar, and S. Koenig, "Multi-agent path finding with mutex propagation," *Artificial Intelligence*, vol. 311, p. 103766, 2022.
- [58] S.-H. Chan, R. Stern, A. Felner, and S. Koenig, "Greedy priority-based search for suboptimal multi-agent path finding," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 16, pp. 11–19, 2023.
- [59] G. Gange, D. Harabor, and P. J. Stuckey, "Lazy cbs: implicit conflict-based search using lazy clause generation," in *Proceedings of the international conference on automated planning and scheduling*, vol. 29, pp. 155–162, 2019.
- [60] Q. Xu, J. Li, S. Koenig, and H. Ma, "Multi-goal multi-agent pickup and delivery," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9964–9971, IEEE, 2022.
- [61] T. T. Walker, N. R. Sturtevant, A. Felner, H. Zhang, J. Li, and T. S. Kumar, "Conflict-based increasing cost search," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 31, pp. 385–395, 2021.
- [62] A. Andreychuk, K. Yakovlev, P. Surynek, D. Atzmon, and R. Stern, "Multi-agent pathfinding with continuous time," *Artificial Intelligence*, vol. 305, p. 103662, 2022.
- [63] I. Solis, J. Motes, R. Sandström, and N. M. Amato, "Representation-optimal multi-robot motion planning using conflict-based search," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4608–4615, 2021.
- [64] X. Zhang, G. Xiong, Y. Wang, S. Teng, and L. Chen, "D-pbs: Dueling priority-based search for multiple nonholonomic robots motion planning in congested environments," *IEEE Robotics and Automation Letters*, 2024.
- [65] J. Chen, J. Li, C. Fan, and B. C. Williams, "Scalable and safe multi-agent motion planning with nonlinear dynamics and bounded disturbances," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, pp. 11237–11245, 2021.
- [66] L.-C. Latombe, "Probabilistic roadmaps for robot path planning," *Practical motion planning in robotics: current approaches and future challenges*, pp. 33–53, 1998.
- [67] "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.
- [68] S. Fortune, "Voronoi diagrams and delaunay triangulations," in *Handbook of discrete and computational geometry*, pp. 705–721, Chapman and Hall/CRC, 2017.
- [69] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "Maprm: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *Proceedings 1999 IEEE international conference on robotics and automation (Cat. No. 99CH36288C)*, vol. 2, pp. 1024–1031, IEEE, 1999.
- [70] F. Aurenhammer, "Voronoi diagrams—a survey of a fundamental geometric data structure," *ACM Computing Surveys (CSUR)*, vol. 23, no. 3, pp. 345–405, 1991.
- [71] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.
- [72] D. O'Sullivan and A. Turner, "Visibility graphs and landscape visibility analysis," *International journal of geographical information science*, vol. 15, no. 3, pp. 221–237, 2001.
- [73] U. Brandes, "A faster algorithm for betweenness centrality," *Journal of mathematical sociology*, vol. 25, no. 2, pp. 163–177, 2001.



**Hannah Lee** is a Software Engineer at Anduril Industries, where she works on autonomy and motion planning. She received her Ph.D. in Computer Science from the University of Illinois Urbana Champaign (UIUC), where this research was conducted, and her B.S. in Computer Science from the Colorado School of Mines. Her research interests include multi-agent pathfinding, multi-robot planning, and artificial intelligence.



**James D. Motes** is a Postdoctoral Researcher in the Siebel School of Computing and Data Science at the University of Illinois at Urbana Champaign (UIUC) where he also completed his Ph.D. in Computer Science. He received both a B.S. in Computer Engineering and a M.S. degree in Computer Science from Texas A&M University. His research interests include task and motion planning, multi-robot systems, and autonomous factories.



**Marco Morales Aguirre** is an Associate Professor of Computer Science at Instituto Tecnológico Autónomo de México (ITAM) and a Teaching Associate Professor in the Siebel School of Computing and Data Science at the University of Illinois Urbana-Champaign. He is a National Researcher (Level II) in the National System of Researchers of Mexico and a member of the Mexican Academy of Computing. He has been a Visiting Professor at Texas A&M University (TAMU) and a Lecturer at Universidad Nacional Autónoma de México (UNAM). He holds a Ph.D. in Computer Science from TAMU, an M.S. in Electrical Engineering, and a B.S. in Computer Engineering from UNAM. He has research interests in motion planning and control for autonomous robots, artificial intelligence, machine learning, and computational geometry.



**Nancy M. Amato** (F'10) is Director of the Siebel School of Computing and Data Science and Abel Bliss Professor of Engineering at the University of Illinois at Urbana Champaign. She received undergraduate degrees in Mathematical Sciences and Economics from Stanford, and M.S. and Ph.D. degrees in Computer Science from UC Berkeley and the University of Illinois, respectively. Before returning to her alma mater in 2019, she was Unocal Professor and Regents Professor at Texas A&M University and Senior Director of Engineering Honors Programs. Amato is known for contributions to robotics task and motion planning, computational biology and geometry, and parallel and distributed computing. Her honors include the inaugural Robotics Medal in 2023, the 2019 IEEE RAS Saridis Leadership Award in Robotics and Automation, the inaugural 2014 NCWIT Harrold/Notkin Research and Graduate Mentoring Award, and the 2014 CRA Haberman Award for contributions to increasing diversity in computing. She is a Fellow of the AAAI, AAAS, ACM, and IEEE.