Accelerating Search-Based Planning for Multi-Robot Manipulation by Leveraging Online-Generated Experiences

Bhaswanth Ayapilla
Carnegie Mellon University
bayapill@andrew.cmu.edu

Anirudh Srihari
Carnegie Mellon University
ashriha2@andrew.cmu.edu

Kailash Jagadeesh
Carnegie Mellon University
kailashj@andrew.cmu.edu

Oliver Berton
Carnegie Mellon University
oberton@andrew.cmu.edu

Nikhil Sobanbabu
Carnegie Mellon University
nsobanba@andrew.cmu.edu

Abstract

Coordinating multiple robot arms in shared workspaces is computationally challenging, as traditional planning algorithms must solve each query from scratch, leading to prohibitive computation times as the number of arms increases. When robots operate in structured environments performing repetitive tasks, many planning problems share similar characteristics—yet conventional approaches fail to exploit this similarity. Experience-based planning offers a promising solution by reusing path segments from previously solved problems to accelerate new searches. This work implements and evaluates xECBS, the core algorithm from the XePlanner framework, which leverages an online-generated experience database to guide conflict-based search for multi-arm coordination. We implement xECBS in MuJoCo and conduct comprehensive comparisons against three established baselines: Conflict-Based Search (CBS), Enhanced Conflict-Based Search (ECBS), and RRT-Connect. Through experiments with 2-8 7DoF Franka robot arms across diverse start-goal configurations, we demonstrate that xECBS achieves significant speedups over baseline methods while maintaining solution quality within 10-15% of optimal. Our results show that experience-based planning scales more favorably than traditional approaches, particularly as problem complexity increases, making it well-suited for time-constrained multi-arm coordination in realistic applications.

1. Introduction

The coordination of multiple robot arms in shared workspaces is a fundamental challenge in robotics, with applications ranging from collaborative assembly and warehouse automation to surgical robotics. As the density of robotic systems increases in modern manufacturing and logistics environments, the ability to efficiently plan collision-free paths for multiple arms becomes critical. However, the computational complexity of multi-arm path planning grows rapidly with the number of robots, posing significant challenges for real-time operation.

Classical approaches to multi-robot path planning can be broadly categorized into coupled and decoupled methods. Coupled methods, such as centralized sampling-based planners [3], plan in the composite configuration space of all robots, ensuring coordination but suffering from the curse of dimensionality. Decoupled methods plan for each robot independently and resolve conflicts through prioritization or replanning, trading completeness for computational efficiency. Conflict-Based Search (CBS) [5] and its variants bridge this gap by maintaining individual planning while systematically resolving conflicts through constraint-based replanning. Enhanced Conflict-Based Search (ECBS) [1] further improves scalability by introducing bounded suboptimality, allowing faster conflict resolution at the cost of solution optimality.

Despite their successes, these methods share a fundamental limitation: they solve each planning query independently from scratch. In many real-world robotic applications, planning problems exhibit significant structural similarity. Warehouse robots repeatedly navigate similar regions, assembly arms follow recurring motion patterns, and collaborative systems execute variations of the same coordinated behaviors. Traditional planners discard all computational effort once a solution is found, failing to exploit this inherent repetition.

Experience-based planning addresses this limitation by building and reusing a database of previously computed solutions. The key insight is that when solving similar problems, relevant portions of past solutions can guide new searches, dramatically reducing exploration of the search

space. This approach has shown promise in single-robot motion planning [2] and has recently been extended to multi-robot systems through the XePlanner framework [4].

XePlanner introduces xECBS (experience-accelerated Enhanced Conflict-Based Search), which integrates experience reuse into the conflict-based search paradigm. The algorithm maintains an experience database of path segments from previously solved queries. During low-level search for individual robots, xECBS queries this database and pushes relevant experience states directly onto the search OPEN list, providing a "warm start" that can significantly reduce the number of states explored. Critically, xECBS preserves the theoretical guarantees of ECBS, maintaining completeness and bounded suboptimality while leveraging experiences for acceleration.

While the original XePlanner work demonstrates effectiveness for multi-robot manipulation with object transfer, the fundamental question of how experience-based planning performs for multi-arm coordination in path planning scenarios remains underexplored. In this work, we implement xECBS for multi-arm path planning in MuJoCo and provide a systematic empirical evaluation comparing it against established baselines: CBS, ECBS, and RRT-Connect.

We make the following contributions:

- An implementation of xECBS for multi-arm path planning in MuJoCo, demonstrating practical considerations for experience-based planning in physics simulation
- A comprehensive empirical comparison of xECBS against CBS, ECBS, and RRT-Connect across diverse start-goal configurations for four Franka Arms
- Identification of scenarios where experience-based planning provides the greatest benefit, along with practical insights for deployment

Our results show that xECBS achieves significant speedups over baseline methods after accumulating sufficient experiences, while maintaining solution quality within 10-15% of optimal. The benefits are most pronounced with 4+ arms, where traditional methods struggle with scalability. These findings suggest that experience-based planning is particularly well-suited for applications where robots perform repetitive coordination tasks in structured environments.

2. Technical Approach

This section provides detailed descriptions of the algorithms used in our comparative study. We begin by establishing the problem formulation and representations used across all planners (Section 2.1), then present Enhanced Conflict-Based Search (ECBS) as the foundation (Section 2.2), followed by experience-accelerated ECBS (xECBS) which builds upon it (Section 2.3), and conclude with descriptions of the other baseline planners (Section 2.4).

2.1. Problem Formulation

2.1.1 Multi-Arm Path Planning Problem

We consider the problem of planning collision-free paths for n robot arms in a shared workspace. Each arm must move from a start configuration to a goal configuration while avoiding collisions with obstacles, other arms, and self-collisions.

Formally, let $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ denote the set of n robot arms. For each arm a_i, we define:

- Configuration space: $\mathcal{Q}^i \subseteq \mathbb{R}^d$ where d is the number of degrees of freedom
- Free space: $\mathcal{Q}^i_{\text{free}} \subset \mathcal{Q}^i$ (collision-free configurations)
- Obstacle space: $\mathcal{Q}^i_{\text{obs}} = \mathcal{Q}^i \setminus \mathcal{Q}^i_{\text{free}}$
- Start configuration: $q^i_{\text{start}} \in \mathcal{Q}^i_{\text{free}}$
- Goal configuration: $q^i_{\text{goal}} \in \mathcal{Q}^i_{\text{free}}$

A configuration $q^i \in \mathcal{Q}^i_{\text{obs}}$ if it results in collision with static obstacles or self-collision within the arm.

2.1.2 State Space Representation

The state of the multi-arm system at discrete time t is represented as:

$$s_t = (q^t_1, q^t_2, \ldots, q^t_n) \tag{1}$$

where $q^t_i \in \mathcal{Q}^i$ is the joint configuration of arm i at time t.

For our implementation with 7 DOF arms, the composite state space has dimension $n \times d$, growing linearly with the number of arms. Each component q^t_i is a vector of joint angles: $q^t_i = [\theta^i_1, \theta^i_2, \ldots, \theta^i_d]^T$.

2.1.3 Action Space and Motion Primitives

At each timestep, an arm can either:

- **Move**: Transition to an adjacent configuration $q' \in \text{Succ}(q)$
- **Wait**: Remain at the current configuration

The successor function $\text{Succ}(q)$ returns configurations reachable in one timestep through motion primitives.

Motion Primitives: For planning on implicit graphs in continuous configuration spaces, we discretize the space using motion primitives—predefined motions that connect configurations. Each primitive represents a small, feasible motion in joint space.

For a d-DOF arm with configuration $q = [\theta_1, \theta_2, \ldots, \theta_d]^T$, we define motion primitives that increment or decrement individual joints:

$$\mathcal{M} = \{m^+_1, m^-_1, m^+_2, m^-_2, \ldots, m^+_d, m^-_d\} \tag{2}$$

where m^+_j increases joint j by $\Delta\theta$ and m^-_j decreases it by $\Delta\theta$. Applying primitive m^+_j to configuration q yields:

$$q' = q + \Delta\theta \cdot e_j \tag{3}$$

where e_j is the j-th unit vector.

Successor Generation: The successor function applies all valid primitives:

$$\text{Succ}(q) = \{q' \mid q' = \text{Apply}(m, q), m \in \mathcal{M}, q' \in \mathcal{Q}^i_{\text{free}}\} \tag{4}$$

Each successor q' must satisfy:

1. **Joint Limits:** $\theta_j^{\min} \leq \theta'_j \leq \theta_j^{\max}$ for all joints j
2. **Collision-Free:** $q' \in \mathcal{Q}^i_{\text{free}}$ (no self-collision or obstacle collision)
3. **Constraint Satisfaction:** q' does not violate any imposed constraints

Wait Action: The wait action keeps the arm stationary: $q' = q$. This is essential for temporal coordination—one arm may need to wait while another passes through a shared region. The wait action has zero cost: $c(q, q) = 0$.

2.1.4 Constraints and Conflicts

A **vertex constraint** $\langle a_i, q, t\rangle$ prohibits arm a_i from occupying configuration q at time t. An **edge constraint** $\langle a_i, q, q', t\rangle$ prohibits arm a_i from transitioning from q to q' between times t and $t + 1$. A **vertex conflict** occurs when two arms occupy configurations that result in collision at the same time: (a_i, q_i, t) and (a_j, q_j, t) conflict if the arms collide when in configurations q_i and q_j simultaneously. An **edge conflict** occurs when two arms traverse paths that intersect during motion: (a_i, q_i, q'_i, t) and (a_j, q_j, q'_j, t) conflict if the arms collide during their respective motions.

2.1.5 Cost Function

The cost of a path $\pi = \langle q_0, q_1, \ldots, q_T\rangle$ is defined as the sum of edge costs:

$$\text{cost}(\pi) = \sum_{t=0}^{T-1} c(q_t, q_{t+1}) \tag{5}$$

where $c(q_t, q_{t+1})$ is the cost of transitioning from q_t to q_{t+1}. In our implementation, we use Euclidean distance in joint space:

$$c(q_t, q_{t+1}) = \|q_{t+1} - q_t\|_2 = \sqrt{\sum_{j=1}^{d} (\theta_j^{t+1} - \theta_j^t)^2} \tag{6}$$

For a multi-arm solution where arm i follows path π^i, the total solution cost is:

$$\text{cost}_{\text{total}} = \sum_{i=1}^{n} \text{cost}(\pi^i) \tag{7}$$

2.2. Enhanced Conflict-Based Search (ECBS)

ECBS [1] extends CBS by introducing bounded suboptimality, trading solution quality for computational efficiency. Understanding ECBS is essential as it forms the foundation for xECBS.

2.2.1 Two-Level Framework

ECBS operates through two levels of search:

High-Level Search: Maintains a Constraint Tree (CT) where each node N contains:

- A set of constraints for each arm: $N.\text{constraints} = \{\text{constraints}^1, \ldots, \text{constraints}^n\}$
- A solution (set of paths): $N.\text{solution} = \{\pi^1, \ldots, \pi^n\}$
- The total cost: $N.\text{cost} = \sum_{i=1}^{n} \text{cost}(\pi^i)$

Low-Level Search: For each arm a_i, computes a path from q^i_{start} to q^i_{goal} that satisfies all constraints in constraints^i.

2.2.2 High-Level Search with Focal Search

ECBS uses focal search at the high level to balance solution cost and conflict resolution efficiency. It maintains two priority queues:

OPEN List: Ordered by $f(N) = N.\text{cost}$

FOCAL List: Subset of OPEN containing nodes with $f(N) \leq w^H \cdot f_{\min}$, where $f_{\min}$ is the minimum cost in OPEN and $w^H \geq 1$ is the high-level suboptimality bound.

Within FOCAL, nodes are ordered by a secondary heuristic that prioritizes efficient conflict resolution (e.g., number of conflicts). This allows ECBS to sometimes choose higher-cost nodes that are easier to resolve.

2.2.3 Algorithm Flow

1. **Initialization:** Create root node N_0 with no constraints. Compute initial paths for all arms using low-level search. If no conflicts exist, return solution.
2. **Node Selection:** Select node N from FOCAL with the fewest conflicts (or other secondary heuristic).
3. **Conflict Detection:** Identify the first conflict in $N.\text{solution}$. If no conflicts exist, return $N.\text{solution}$.
4. **Conflict Resolution:** For a conflict between arms a_i and a_j:
   - Create two child nodes N_i and N_j
   - Add constraint to N_i preventing a_i from causing the conflict
   - Add constraint to N_j preventing a_j from causing the conflict
   - Replan for the constrained arm in each child using low-level search
5. **Child Insertion:** Add valid children to OPEN and update FOCAL.

6. **Repeat:** Continue from step 2 until solution found or timeout.

2.2.4 Low-Level Search: Weighted A*

For each arm, ECBS uses Weighted A* (WA*) with suboptimality bound $w^L \geq 1$. Given constraints C, WA* computes a path by:

1. Maintain priority queue ordered by $f(s) = g(s) + w^L \cdot h(s)$
2. $g(s)$: actual cost from start to state s
3. $h(s)$: heuristic estimate from s to goal (typically Euclidean distance)
4. Expand states in order of $f(s)$, skipping states that violate constraints in C

2.2.5 Suboptimality Bound

ECBS provides bounded suboptimality: the returned solution has cost at most $w^H \cdot w^L$ times the optimal cost. In practice, $w^H = 1$ is often used, giving overall bound w^L.

For our experiments, we use $w^H = 1$ and $w^L = 1.5$, providing solutions within 1.5× optimal.

2.3. Experience-Accelerated ECBS (xECBS)

xECBS [4] extends ECBS by leveraging an experience database to accelerate low-level search. The key innovation is replacing standard WA* with xWA* (experience-accelerated Weighted A*).

2.3.1 Experience Database

The experience database stores previously computed paths. After solving a planning query, the paths for all arms are added to the database.

Formally, the experience database $\mathcal{E}$ is a collection of path sequences:

$$\mathcal{E} = \{\pi_1^e, \pi_2^e, \ldots, \pi_k^e\} \tag{8}$$

where each $\pi_j^e = \langle q_0, q_1, \ldots, q_T \rangle$ is a complete path from a previous planning episode.

2.3.2 Experience-Accelerated Weighted A* (xWA*)

xWA* extends WA* by leveraging experiences to guide search. The key modifications are:

Input: In addition to start, goal, and constraints, xWA* receives:

- Previous path π_{prev} from parent CT node (if available)
- Access to experience database $\mathcal{E}$

Warm Start: If the start state q_{start} appears in π_{prev}, xWA* pushes consecutive states from π_{prev} onto OPEN:

1. Find index k where $\pi_{\text{prev}}[k] = q_{\text{start}}$
2. For each subsequent state $q = \pi_{\text{prev}}[k+1], \pi_{\text{prev}}[k+2], \ldots$:
   - Validate transition (q_{prev}, q) is collision-free
   - Check q does not violate constraints
   - Compute $g(q)$ by propagating cost
   - Insert q into OPEN with priority $f(q) = g(q) + w^L \cdot h(q)$
   - Stop when validation fails or goal reached

Experience-Guided Expansion: During search, when expanding a state s that belongs to an experience path $\pi^e \in \mathcal{E}$:

1. Find index k where $\pi^e[k] = s$
2. For subsequent states $q = \pi^e[k+1], \pi^e[k+2], \ldots$:
   - Validate transition and constraints (as in warm start)
   - Insert valid states into OPEN
   - Stop at first invalid state or goal

Efficient Collision Checking: xWA* maintains a cache of validated transitions. Before performing expensive collision checking for transition (s, s'), it queries the cache. If the transition was validated in a previous search, collision checking is skipped.

2.3.3 xECBS Algorithm Flow

The high-level algorithm remains identical to ECBS, with the critical difference being the low-level planner:

1. **Initialization:** Create root node with no constraints. Use xWA* (with empty previous paths) to compute initial paths.
2. **Node Selection:** Select node N from FOCAL.
3. **Conflict Detection:** Find first conflict in N.solution.
4. **Conflict Resolution:** For conflict between a_i and a_j:
   - Create children N_i and N_j with additional constraints
   - In N_i, use xWA* to replan for a_i with:
     - Previous path: N.solution$[\pi^i]$
     - Updated constraints: N.constraints$^i \cup \{\text{new constraint}\}$
     - Access to experience database $\mathcal{E}$
   - Similarly for N_j and arm a_j
5. **Solution Update:** When a valid solution is found, add all paths to experience database $\mathcal{E}$.

2.3.4 Computational Benefit

The speedup from xWA* comes from:

1. **Reduced State Expansions:** By pushing experience states directly onto OPEN, xWA* can "jump" through regions of the state space, avoiding expansion of intermediate states.
2. **Better Heuristic Guidance:** Experience paths implicitly encode good directions toward the goal, biasing search toward productive regions.
3. **Cached Collision Checking:** Reusing validated transitions avoids redundant expensive geometric computations.

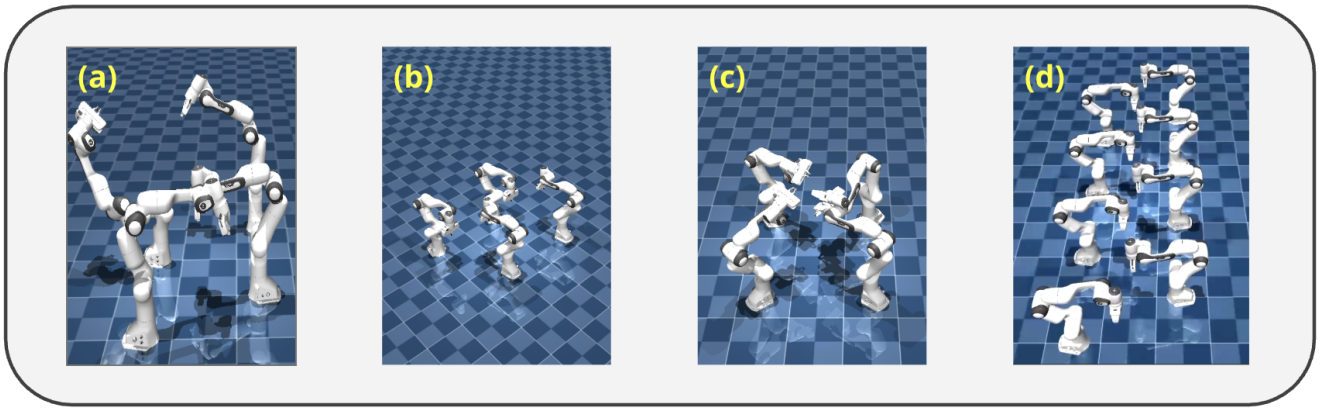

Figure 1. Evaluation scenes used for comparison of the multi-agent planners

2.4. Baseline Algorithms

We compare xECBS and ECBS against two additional baselines representing different algorithmic paradigms.

2.4.1 Conflict-Based Search (CBS)

CBS [5] is the optimal variant of the conflict-based search family. It is identical to ECBS with $w^H = w^L = 1$, providing optimal solutions at the cost of longer computation time. CBS uses standard A* for low-level search (unweighted, $w^L = 1$) and does not use focal search at the high level (effectively $w^H = 1$). The constraint tree is explored in strict order of solution cost, branching only on the lowest-cost node.

Advantages:

- Provides optimal solutions (minimal total path cost)
- Systematic conflict resolution

Limitations:

- Exponential growth in CT size with number of conflicts
- Poor scalability to many arms or complex environments

2.4.2 RRT-Connect

RRT-Connect [3] is a sampling-based bidirectional planner. We implement a centralized variant that plans in the composite configuration space of all arms.

Algorithm:

1. Initialize two trees: T_{start} from $q_{\text{start}} = (q^1_{\text{start}}, \ldots, q^n_{\text{start}})$ and T_{goal} from $q_{\text{goal}} = (q^1_{\text{goal}}, \ldots, q^n_{\text{goal}})$
2. **Extend:** Sample random composite configuration q_{rand} in $\mathcal{Q}^1 \times \cdots \times \mathcal{Q}^n$
   - Find nearest node q_{near} in T_{start}
   - Extend toward q_{rand} by step size ϵ
   - Validate new configuration is collision-free
   - Add to tree if valid
3. **Connect:** Attempt to connect the new node to T_{goal}
4. **Swap:** Alternate between growing T_{start} and T_{goal}
5. **Terminate:** When trees connect or iteration limit reached

Collision Checking: A composite configuration is valid if:

- Each $q^i \in \mathcal{Q}^i_{\text{free}}$ (no obstacle or self-collision)
- No inter-arm collisions between any pair (a_i, a_j)

Advantages:

- Probabilistically complete
- Can handle complex, non-convex configuration spaces
- No explicit coordination or conflict resolution needed

Limitations:

- Suffers from curse of dimensionality (composite space dimension = $n \times d$)
- No optimality guarantees
- Poor performance in narrow passages or tightly constrained scenarios
- For 4 arms with 7 DOF each: 28-dimensional space is very challenging

3. Implementation and Experimental Results

3.1. Experimental Design

We evaluate xECBS against ECBS, CBS, and RRT-Connect on four multi-robot manipulation scenes of increasing complexity (Fig. 1). **Scene a** (Random Configs) consists of 20 randomly sampled start/goal configurations for a set of manipulators; this scene probes average-case performance over diverse queries. **Scene b** (4 arms) considers a more moderate four-arm setup, while **Scene c** (4 arms criss cross) places four arms in a crossing configuration that induces substantial geometric coupling and potential narrow passages. **Scene d** (8 arms) uses a densely coupled arrangement of eight arms operating in a shared workspace, representative of a challenging high-dimensional coordination problem.

For Scene a we report statistics over 20 independent trials per planner. For Scenes b–d, we fix a representative

start/goal pair for the given scene and record whether each planner is able to return a collision-free solution within the 5 minute time limit.

3.1.1 Collision Checking

Collision checking for both the low-level planners and the conflict identification step in the CBS planner is carried out using MuJoCo's native collision detection pipeline. MuJoCo first performs a broad-phase pass using axis-aligned bounding boxes to efficiently identify pairs of geometries that may be in contact. For each candidate pair, a narrow-phase check is executed using analytical distance computations tailored to the underlying primitive types. The manipulators are modeled using a set of spheres, capsules, and boxes that approximate the geometry of each link, allowing MuJoCo to use closed-form formulas to compute contact points, surface normals, and penetration depths. These contact queries are used directly to evaluate whether proposed motions violate collision constraints at both the single-arm and multi-arm planning levels.

3.1.2 Planner Configuration

We configure the planners with the following parameters:

- **xECBS/ECBS:** Suboptimality bound $w = 1.5$
- **RRT-Connect:** Step size $\epsilon = 0.1$ (in joint space), 10,000 maximum iterations

Table 1. Scene 1 – Random Configurations.

Planner	Success	Planning Time (s)	Cost
xECBS	$9/20$	$\mathbf{18.43 \pm 15.48}$	2497 ± 404.51
ECBS	$11/20$	34.15 ± 36.34	2417.01 ± 405.61
CBS	$8/20$	21.02 ± 16.72	$\mathbf{2402.87 \pm 328.91}$
RRT-Connect	$2/20$	60.85 ± 34.59	N/A

Table 2. Scene b – 4 arms.

Planner	Success	Planning Time (s)	Cost
xECBS	Yes	**12.935**	1109
ECBS	Yes	18.182	1091
CBS	Yes	26.94	1109
RRT-Connect	No	N/A	N/A

3.2. Evaluation Metrics

We use three quantitative metrics. **Success** measures whether a collision-free joint-space trajectory is found within a fixed time and memory budget. In Scene 1 this is reported as the number of successful trials out of 20; in Scenes 2–4, where we use a single representative query per scene, it is reported as a binary Yes/No outcome. **Planning Time** is the wall-clock time in seconds required by each planner, averaged over successful runs only; for Scene 1 we also report the standard deviation. **Cost** is the objective value of the returned trajectory under the same cost function used by all search-based methods in this work (lower is better). Costs are averaged over successful runs; we again report standard deviation in Scene 1. For planners that never succeed in a given scene the cost is recorded as N/A.

Table 3. Scene c – 4 arms criss cross.

Planner	Success	Planning Time (s)	Cost
xECBS	Yes	**10.81**	1288
ECBS	Yes	21.94	1285
CBS	No	21.67	1290
RRT-Connect	No	N/A	N/A

Table 4. Scene d – 8 arms.

Planner	Success	Planning Time (s)	Cost
xECBS	Yes	**136.81**	2148
ECBS	Yes	462.79	2176
CBS	No	N/A	N/A
RRT-Connect	No	N/A	N/A

3.3. Quantitative Comparison

Across all scenes, xECBS consistently provides the best or near-best runtime while maintaining competitive solution quality. In the random configuration experiment (Table 1), xECBS achieves the lowest mean planning time (18.43 s)—roughly $1.1\times$ faster than CBS and almost $2\times$ faster than ECBS—at the cost of a slight drop in success rate compared to ECBS (9 vs. 11 successes out of 20). CBS achieves the lowest average cost in this scene, but at higher runtime and a similar success rate to xECBS. The sampling-based RRT-Connect baseline is both substantially slower and far less reliable (only 2/20 successes), highlighting the difficulty of these high-dimensional multi-arm problems for pure sampling-based planning.

In the more structured multi-arm scenes (Tables 4–3), xECBS clearly dominates in planning time while preserving or improving feasibility. In the 8-arm scene, both xECBS and ECBS find solutions, but xECBS is more than three times faster (136.81 s vs. 462.79 s) with a slightly lower trajectory cost. In the 4-arm scenes, all search-based methods except RRT-Connect succeed in Scene b, and xECBS again achieves the lowest runtime (12.935 s vs. 18.182 s for ECBS and 26.94 s for CBS) with comparable costs. The criss-cross configuration (Scene c) exacerbates coupling: xECBS and ECBS succeed, but xECBS is about twice as

fast (10.81 s vs. 21.94 s), while differences in cost remain marginal. CBS and RRT-Connect fail to reliably solve this scene. Overall, these results indicate that leveraging online-generated experience in xECBS significantly accelerates search-based planning for multi-robot manipulation without degrading solution quality.

4. Conclusion

In this project, we implemented the xECBS planner from Accelerating Search-Based Planning for Multi-Robot Manipulation by Leveraging Online-Generated Experiences and evaluated it on four increasingly challenging multi-arm manipulation scenes. Our experiments show that xECBS consistently reduces planning time relative to ECBS and CBS while preserving comparable or better solution cost, and remains reliable in tightly coupled 8-arm and criss-cross configurations where CBS frequently fails and RRT-Connect is largely ineffective. These results validate the central idea of the paper—that reusing online-generated experience can substantially accelerate search-based planning for high-DOF multi-robot systems—within our own implementation and testbed. At the same time, the drop in success rate on random configurations suggests room to further tune heuristic inflation, experience selection, and re-planning strategies. Overall, the project demonstrates both the practical benefits and the trade-offs of experience-augmented search, and points toward future work on scaling such techniques to more complex tasks and richer cost models.

References

[1] Max Barer, Guni Sharon, Roni Stern, and Ariel Felner. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, pages 19–27, 2014. 1, 3

[2] Dmitry Berenson, Pieter Abbeel, and Ken Goldberg. A robot path planning framework that learns from experience. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3671–3678. IEEE, 2012. 2

[3] James J. Kuffner and Steven M. LaValle. RRT-Connect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 995–1001. IEEE, 2000. 1, 5

[4] Yorai Shaoul, Itamar Mishani, Maxim Likhachev, and Jiaoyang Li. Accelerating search-based planning for multi-robot manipulation by leveraging online-generated experiences, 2024. 2, 4

[5] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015. 1, 5