Real-Time Multi-Robot Motion Planning with Safe-Interval Search and Learning-Guided Repair

Rajat Kumar* , **Kristin Predeck** , **Ken Meszaros** and **Trevor Dardik**
Amazon.com Services LLC.
*kmardpl@amazon.com

Abstract

Motion planning among multiple robots in a shared space is a fundamental yet computationally challenging problem in robotics, with applications ranging from warehouse automation to autonomous fleets. In this work, we introduce a fast, scalable motion planner that achieves real-time, collision-free trajectory planning via a two-staged algorithm combining deterministic search-based planning with machine learning-driven conflict resolution. We present a prioritized Safe Interval Path Planning algorithm (SIPP-PP) with a novel limited goal reservation strategy to prevent goal-blocking conflicts while allowing shared goal regions. We added a second layer of ML-guided Large Neighborhood Search (LNS) procedure to our SIPP-PP algorithm for improving success rates in highly congested environments via intelligent selection of conflict resolution actions. The result is a planning system that generates collision-free paths for multiple robots in complex environments within tens of milliseconds. For example, compared to recent advanced learning-based methods such as diffusion planners, our planner is two-to-three orders of magnitude faster. Our work demonstrates a multi-robot planner capable of real-time operation in dense scenarios, satisfying the stringent requirements of industrial applications such as drive units in fulfillment centers.

1 Introduction

Multi-robot motion planning (MRMP) [Bui, 2023] is the problem of computing collision-free trajectories for a fleet of robots operating in a shared environment. For example, fulfillment centers may deploy thousands of robots for material handling within a warehouse [Wurman *et al.*, 2008; Agaskar *et al.*, 2025]. Such settings demand finding collision-free paths for each robot in presence of static obstacles and other robots. Not only that, successful deployment in industrial settings will often require meeting time constraints to meet system's throughput demands. MRMP is a NP-hard problem [Hopcroft and Wilfong, 1986; Yu and LaValle, 2013]. This is often demonstrated by coupled planning algorithms, which treat all robots as a single composite system in a high dimensional space [Schwartz and Sharir, 1983]. As the combined state space grows exponentially with a large number of robots, planning optimal, collision free path becomes computationally intractable. One way to mitigate the computational limitations is via using decoupled approaches like prioritized planning [Erdmann and Lozano-Pérez, 1987]. However, while decoupled approaches are fast, they sacrifice completeness or optimality for efficiency and fail in congested environments where robot paths inherently conflict between start and goal locations. Recent work on using generative models (e.g. diffusion models) for multi-robot trajectory data has shown utility in generating smooth, human-like coordinated motions [Carvalho *et al.*, ; Shaoul *et al.*, 2025]. However, such methods currently suffer from large computation time and do not guarantee finding a solution in a fixed time interval. For example, the state-of-the-art Multi-Robot Motion Diffusion (MMD) method [Shaoul *et al.*, 2025] required 10-15 seconds to plan for 9 robots; and struggled in congested environments (50% success for 9 robots in a bottleneck scenario). This gap between real-time deployment requirements in industry and the capabilities of existing planners motivates our current work.

In this paper, we propose a new planning algorithm that utilizes the reliability of classic planners and the efficiency associated with learned approaches to achieve fast, scalable multi-robot motion planning. In particular, we propose a two-stage planning algorithm, wherein, a fast deterministic planner is used to generate an initial set of conflict-free paths; followed by a machine learning (ML) driven refinement phase to improve solution quality and handle residual conflicts. Our algorithm is built on the following pieces:

1. **Safe-Interval Prioritized Planning (SIPP-PP):** Safe Interval Path Planning (SIPP) [Phillips and Likhachev, 2011] is a well known algorithm for navigating dynamic environments via searches in a combined space-time domain and exploiting contiguous "safe intervals" to reduce state explosion. We adapt SIPP to the multi-robot domain using prioritized planning, wherein each robot plans individually in sequence, avoiding higher-priority robots' trajectories. Importantly, we introduce a limited goal reservation technique that reserves each robot's goal for a bounded time window after arrival, preventing

immediate collisions while allowing robots with nearby goals to coexist. As a result, we obtain a method that eliminates a common failure mode of decoupled planners, and we call this Safe-Interval Prioritized Planning (SIPP-PP).

2. **XGBoost-Guided Large Neighborhood Search (LNS):** Even though SIPP-PP yields a valid solution in most cases, heavily congested environments, such as multiple robots facing deadlock in narrow corridors or cyclic swapping situations that a fixed priority ordering cannot resolve, will degrade task success for SIPP-PP. To address this, we add to our SIPP planner a secondary phase of Large Neighborhood Search (LNS) [Shaw, 1998]. The LNS iteratively repairs the solution by re-planning a subset of robots. We make this LNS phase learning-guided. Instead of using manually designed heuristics to choose which robots to replan, we train an XGBoost model to pick the best robots for re-planning to achieve collision-free paths with minimal effort and path length.

We call this combined system SIPP-PP-LNS. This system demonstrates near-perfect success rates for moderate robot densities and slow performance degradation rate at high densities; while always maintaining low planning times, generally on the order of 1-10 milliseconds and at most a couple of seconds for extremely challenging environments.

Contributions.

1. A fast and deterministic multi-robot planner based on safe-interval search and prioritized planning, enhanced with a novel goal-time reservation strategy that significantly boosts success rates;
2. A machine learning-guided repair algorithm that improves solution quality and success in congested environments by intelligently selecting re-planning interventions;
3. Discussion on how our method's real-time performance makes it well-suited for industrial deployment in settings such as warehouse automation.

We organize the rest of the paper as follows. In Section 2, we formally define the motion planning problem along with the relevant background/prior work. Section 3 details our approach, provides algorithmic details for the SIPP-PP algorithm and the XGBoost-guided LNS strategy, and provides discussions of design choices. Section 4 provides the experimental setup and results and finally, Section 5 concludes the paper.

2 Problem Formulation and Background

2.1 Multi-Robot Motion Planning (Problem Definition)

We start with n robots operating in a 2-dimensional space $W \subset \mathbb{R}^2$ with static obstacles. The goal of Multi-Robot Motion Planning (MRMP) is to compute collision-free trajectories for each robot in order to optimize a given objective function. For modeling the process, we start by assuming each robot i as a disc of radius r that moves with holonomic dynamics i.e., the robot can move in any direction in the plane and bounded speed. Let $s_i \in W$ and $g_i \in W$ denote the start and goal positions of the robot i respectively. Then, we can define the goal as — compute a trajectory $\tau_i : [0, T] \to W$ for each robot i from s_i to g_i within a time horizon T such that the following conditions are satisfied for all robots i and all times $t \in [0, T]$:

a) The Boundary Condition: We define that each robot starts at its start location $\tau_i(0) = s_i$ with a target to reach its goal location by the end of time horizon, i.e. $\tau_i(T) = g_i$. We further assume that all robots start moving at time 0. If a robot starts later (delayed start), we model that by having τ_i remain at s_i for an initial waiting period;

b) Obstacle Avoidance: We say that for $t \in [0, T]$, $\tau_i(t)$ maintains a clearance from obstacles and each robot's center stays in a free space,i.e., $\text{dist}(\tau_i(t), O) > r$ for every obstacle $O \subset W$ to do obstacle avoidance; and,

c) Inter-robot Collision: Finally, we focus on dynamic entities i.e. no two robot should collide. We say that for any pair of distinct robots $i \neq j$ and at all times t, the distance between robots must exceed a safety threshold: $|\tau_i(t) - \tau_j(t)| > 2r + \epsilon$. Here $2r$ is the minimum distance between centers to avoid body contact (since each has radius r), and $\epsilon > 0$ is an extra safety margin. We adopt a small ϵ (e.g. $0.1r$) to account for numerical tolerance and ensure strict separation. This condition ensures that robots can't swap places at the exact same time (which would entail a distance of exactly $2r$ at the moment they touch). Further, this prevents crowding too closely during motion.

We then define the solution to multi-robot motion planning problem as a set of n trajectories $\tau = \tau_1, \tau_2, ..., \tau_n$ satisfying the above constraints, and these τ are collision-free and feasible. In general for MRMP, the goal is to obtain any feasible solution that minimizes an objective function, such as makespan T (finishing time for all robots) [Sharon *et al.*, 2015; Yu and LaValle, 2016]. In this paper, we prioritize finding any feasible solution quickly over optimizing path lengths [Li *et al.*, 2021; Okumura, 2024]. This is motivated by real-time applications where new tasks continuously arrive and it is more important to immediately find a valid coordinated motion than to find the absolute shortest paths [Agaskar *et al.*, 2025]. We do however incorporate secondary optimization goals like reducing path length and smoothness once a feasible solution is found (in the LNS improvement stage).

2.2 Problem Complexity

Discrete MAPF problem on a grid, with constraints defined in section 2.1, is NP-hard to solve optimally. Methods such as Conflict-Based Search (CBS) guarantee finding the shortest paths but may require exploring an exponential number of conflict resolutions [Sharon *et al.*, 2015]. Given we seek a solution applicable to real-world industrial deployment, we do not attempt an optimal joint search, and rather plan individual paths (decoupled planning) and handle conflicts via additional phases. It is well known that decoupled methods such as prioritized planning are not complete [Erdmann and Lozano-Pérez, 1987; van den Berg and Overmars, 2005], but still widely used in practice due to their speed. Our work enhances a decoupled framework with corrective strategies

(goal reservations, dynamic reordering via LNS) to effectively solve challenging scenarios.

2.3 Background: SIPP, Prioritized Planning, and LNS

Safe Interval Path Planning (SIPP). Safe Interval Path Planning (SIPP) performs efficient planning in dynamic environments by partitioning time at each cell into *safe intervals*—maximal contiguous time spans where the cell is collision-free [Phillips and Likhachev, 2011]. SIPP doesn't expand states at every timestep, and rather searches over (cell, interval) pairs wherein it collapses many timesteps into single states when the cell is free. This reduced state space is the primary reason for fast search with SIPP [Phillips and Likhachev, 2011]. For planning for robot i, all previously planned robots represent blocked intervals through their reserved space-time cells. Aditionally, SIPP tracks edge reservations to prevent swap conflicts.

Prioritized Planning (PP) and Goal Reservation. Prioritized planning (PP) is a decoupled approach which sequentially plans robots in a fixed order using strict priority ordering [Erdmann and Lozano-Pérez, 1987; van den Berg and Overmars, 2005]. PP treats higher-priority robots as dynamic obstacles for lower-priority robots. PP is fast but incomplete in general and can fail in congested environments [LaValle, 2006]. Additionally, a well-known issue with the standard PP is the goal conflict problem [Čáp *et al.*, 2015], wherein a high-priority robot will reserve its goal location for all future times and thus can potentially block lower-priority robots from passing through that location even if it might be safe for them to do so. The converse is true as well, and the deadlock comes from robots blocking each other as robot A's goal is in robot B's path and vice versa. We approach this with a *limited goal reservation* strategy, where the idea is that each robot's goal is reserved for a bounded time window after arrival (e.g., 15 timesteps) rather than permanently. This prevents immediate collisions at goals while allowing later robots to reach nearby or overlapping goal regions, particularly when multiple continuous-space goals discretize to the same grid cell.

Large Neighborhood Search (LNS) for MAPF. LNS-style MAPF solvers generate an initial solution quickly (often via PP) [Shaw, 1998] and then repeatedly destroy/repair subsets of agents to reduce collisions or improve cost within a time budget. Recent anytime MAPF-LNS methods demonstrate strong scalability and fast improvement rates [Li *et al.*, 2021]. Our work follows this paradigm but targets *feasibility under strict time constraints* and introduces a lightweight learning component to guide repair decisions.

Learning-guided search decisions. Rather than learning full trajectories, we learn a small policy that helps choose among discrete repair operators during LNS. We use gradient-boosted decision trees (XGBoost), a robust and efficient model class for structured features [Chen and Guestrin, 2016]. This yields predictable runtime and strong generalization in the low-data regime common in robotics systems engineering.

3 Approach: SIPP-PP with Learning-Guided LNS

3.1 Phase 1: SIPP-PP (Initial Plan)

We employ prioritized planning, which selects an ordering σ of the n robots and plans their paths sequentially. The k-th robot in the order plans in a dynamic environment defined by the reservation table $\mathcal{R}$ of all previously planned robots. We use Safe-Interval Path Planning (SIPP) as the low-level single-agent solver.

Ordering heuristics. We evaluate the following candidate orderings and return the first one that yields a complete feasible plan:

- longest-first: $\sigma = \mathrm{argsort}(-\hat{t}_i)$, where $\hat{t}_i = \|s_i - g_i\|_1$ (Manhattan distance from start to goal),
- shortest-first: $\sigma = \mathrm{argsort}(\hat{t}_i)$,
- original index order,
- random permutation.

This small set of heuristics trades optimality for robustness and speed.

Delayed starts and waiting. If SIPP fails to find a path for robot i under the current reservations, we retry with a delayed start time $d \in \{1, \ldots, T_{\max}\}$, allowing the robot to wait at its start position until earlier conflicts resolve. SIPP itself also permits waiting at any point along the path (not only at the start) to avoid transient conflicts. To ensure safety during the waiting period, the start position s_i of an unplugged robot is treated as a static obstacle in $\mathcal{R}$ for all higher-priority robots up to the time d.

Goal reservation. After successfully planning robot i, we reserve its entire path in the space-time reservation table $\mathcal{R}$ and reserve its goal cell for a *limited time window* after arrival: $(g_i, t) \in \mathcal{R}$ for $t_{\text{arrival}} < t < t_{\text{arrival}} + \Delta$, where Δ is a small buffer (e.g., 15 timesteps). We assumes that agents go off the active navigation grid (e.g., into a workstation or chute) upon task completion, preventing permanent blocking of the goal cell.

3.2 Safe-Interval Path Planning (SIPP)

SIPP operates on a discretized grid where each cell v has an associated set of *safe intervals*—contiguous time ranges $[t_{\text{start}}, t_{\text{end}}]$ during which the cell is unoccupied. These intervals are computed from the reservation table $\mathcal{R}$: a cell is blocked at time t if $(v, t) \in \mathcal{R}$, and safe intervals are the maximal gaps between blocked times.

Each search state is (v, I) where v is a grid cell and I is a safe interval index, with an associated earliest arrival time within that interval. Transitions to a neighbor cell v' are valid if we can depart from v within its current safe interval and arrive at v' within one of v''s safe intervals. Specifically, departing at time t and arriving at $t + 1$ requires:

- t falls within the current interval of v,
- $t + 1$ falls within some safe interval of v',
- no edge conflict exists (swap prevention).

Algorithm 1 SIPP-PP (Prioritized Planning with SIPP)

Initialize empty reservation table $\mathcal{R}$
Compute $\hat{t}_i = \|s_i - g_i\|_1$ for all robots i
Generate candidate orderings: longest-first, shortest-first, original, random
best $\Pi \leftarrow \emptyset$, best num planned $\leftarrow 0$
for each ordering σ **do**
  $\mathcal{R} \leftarrow \emptyset$ {reset reservations for new ordering}
  current $\Pi \leftarrow \emptyset$
  for robot i in order σ **do**
    $\pi_i \leftarrow \text{SIPP}(s_i, g_i, \mathcal{R})$
    if $\pi_i = \emptyset$ **then**
      for $d = 1, \ldots, T_{\max}$ **do**
        $\pi_i \leftarrow \text{SIPP}(s_i, g_i, \mathcal{R}, \text{start\_time} = d)$
        if $\pi_i \neq \emptyset$ **then**
          break
        end if
      end for
    end if
    if $\pi_i = \emptyset$ **then**
      break {ordering failed; try next}
    end if
    Reserve path π_i and goal (g_i, t) for $t_{\text{arrival}} \leq t < t_{\text{arrival}} + \Delta$ in $\mathcal{R}$
    Add π_i to current Π
  end for
  num planned $\leftarrow$ |current Π|
  if num planned $>$ best num planned **then**
    best $\Pi \leftarrow$ current Π
    best num planned $\leftarrow$ num planned
  end if
  if num planned = n **then**
    return best Π {complete plan found}
  end if
end for
return best Π if best num planned > 0 else failure =0

Algorithm 2 XGBoost-Guided LNS (Anytime Repair)

Define: EvaluateCost(Π) = ($N_{\text{unplanned}} \times W_{\text{fail}}$) + CountConflicts($\Pi$)
$\Pi \leftarrow$ result from Algorithm 1
best $\Pi \leftarrow \Pi$, min cost $\leftarrow$ EvaluateCost(Π)
for iteration $\ell = 1, \ldots$ until time budget expires **do**
  if min cost = 0 **then**
    return best Π
  end if
  Extract features ϕ_i {Unplanned robots assigned high-priority features}
  Compute $\psi_i = f_\theta(\phi_i)$
  $D \leftarrow$ top-k robots by ψ_i (decreasing order)
  $\Pi_{prev} \leftarrow \Pi$, $\mathcal{R}_{prev} \leftarrow \mathcal{R}$ {Backup full state}
  Remove reservations of all robots in D from $\mathcal{R}$
  $batch\_success \leftarrow$ true
  for robot $i \in D$ in decreasing order of ψ_i **do**
    $\pi_i^{new} \leftarrow \text{SIPP}(s_i, g_i, \mathcal{R})$ with delayed-start fallback
    if $\pi_i^{new} \neq \emptyset$ **then**
      Reserve π_i^{new} and limited goal in $\mathcal{R}$
      Update Π with π_i^{new}
    else
      $batch\_success \leftarrow$ false; **break**
    end if
  end for
  if $batch\_success$ **then**
    $current\_cost \leftarrow$ EvaluateCost(Π) {Global recheck}
    if $current\_cost < min\_cost$ **then**
      $best\_\Pi \leftarrow \Pi$; $min\_cost \leftarrow current\_cost$
    end if
  else
    $\Pi \leftarrow \Pi_{prev}$; $\mathcal{R} \leftarrow \mathcal{R}_{prev}$ {Rollback to backup}
  end if
end for
return best Π =0

We use A* with $f = g + h$, where g is the arrival time and $h(v) = \|v - g_i\|_1$ (Manhattan distance to goal). By searching over intervals rather than individual timesteps, SIPP collapses many equivalent states, significantly reducing the search space when reservations are sparse.

3.3 Phase 2: XGBoost-Guided Large Neighborhood Search (Anytime Repair)

When prioritized planning produces a plan with residual conflicts or fails to produce a complete feasible plan (or yields a highly suboptimal one), we apply an anytime large neighborhood search (LNS) guided by a learned model.

Features and Learning. For each robot i, we compute a feature vector $\phi_i \in \mathbb{R}^8$. For robots with valid paths from the initial stage, the feature vector consists of normalized conflict count, partner conflict density (average conflict count of robots that i conflicts with), path flexibility ($(|\pi_i| - \|s_i - g_i\|_1)/|\pi_i|$), conflict timing, spatial centrality (proximity of path midpoint to workspace center), goal congestion (fraction of robots with goals within 2 cells of g_i), bottleneck score (fraction of i's path cells shared with other robots), and normalized path length. Crucially, for robots that failed to find a path in the initialization phase, we assign a distinct indicator feature vector to ensure they are prioritized for insertion during the LNS destroy step. We then train an XGBoost regressor on planning traces to predict a joint objective combining conflict resolution and path optimality.

Robot subset selection. At runtime, we predict for each robot a score $\psi_i = f_\theta(\phi_i)$ representing its expected contribution to conflict resolution and path quality. We select the top-k robots with highest ψ_i (default $k = 3$) to form the destroy set D.

Repair procedure. The selected robots are removed from $\mathcal{R}$. Then, they are replanned *sequentially* in decreasing order of ψ_i, using SIPP with delayed-start fallback. Each successfully replanned path is immediately reserved in $\mathcal{R}$ before the next robot in D is planned, ensuring the subset remains conflict-free internally. If SIPP fails for a robot, its previous path is re-reserved in $\mathcal{R}$ to maintain consistency.

3.4 Runtime Considerations

Let $|V|$ be the number of free cells and H the time horizon. A single SIPP search has worst-case complexity $O(|V|H\log(|V|H))$. SIPP-PP performs $O(n)$ searches per ordering across a constant number of orderings, yielding $O(n|V|H\log(|V|H))$. LNS adds up to K iterations (capped by wall-clock time), each replanning $k \ll n$ robots, for $O(Kk|V|H\log(|V|H))$. XGBoost inference costs $O(n)$ per iteration. The overall method is naturally anytime.

4 Experimental Setup and Results

We evaluated our approach on a comprehensive set of scenarios as used in previous work. We aimed at measuring: a) Success rate – the percentage of instances for which a collision-free solution is found within a given time limit (we used 60 seconds as the limit); b) Planning time – the wall-clock time taken to compute a solution; and c) Path length as a secondary quality metric for optimal planning [Stern *et al.*, 2019]. We compare our results against prioritized planning [Erdmann and Lozano-Pérez, 1987], conflict-based search [Sharon *et al.*, 2015] and A*-ECBS [Cohen *et al.*, 2016].

4.1 Setup and Scenarios

Environment maps: We evaluate on four map configurations within a 2×2m workspace ($[-1,1]\times[-1,1]$). We used a robot radius r = 0.05m and $\delta = 2.1r = 0.105$m. We define the safety constraint as $|\tau_i(t) - \tau_j(t)| \geq 2r + \epsilon$, setting $\epsilon = 0.004$m. This ensures strict separation while allowing valid movements between adjacent cells centers (distance 0.105m). This yields a grid size of 21×21. The four map configurations we use are: a) **Empty**: For the empty map, we create a zero obstacle and simple environment, wherein conflicts occur purely from robot-robot interactions. This map serves as a baseline for coordination ability without environmental constraints; b) **Highways**: For highways map, we use a central 0.5×0.5m obstacle surrounded by four 0.25×0.25m corner obstacles, creating a cross-shaped corridor system, wherein the robots are assigned start and goal positions to be in opposite corridors. As a result, all paths converge at the central intersection creating a four-way bottleneck, and allows us to test whether robots from perpendicular directions can coordinate passage via a single narrow crossing; c) **Conveyor**: For the conveyor map, we use three horizontal barriers to partition the workspace into two parallel corridors connected by a central gap. Half the robots travel horizontally within their corridor (right-to-left and left-to-right), while the other half must cross diagonally between corridors through the center gap. This maps allows us to test conflicts between horizontal traffic and perpendicular crossers and simulates bidirectional conveyor lanes with a transfer station; and, d) **Drop Region**: For this, we create four 0.55×0.55m corner obstacles and a central obstacle. This causes a congested workspace with narrow diagonal passages, wherein robots start in one quadrant and their goal location is in diagonally opposite quadrant. As a result, the robots are forced to travel through constrained gaps around the center. Drop region map models package staging areas in a fulfillment center, where drive units need to travel between storage zones and drop chutes through limited access points [Wurman *et al.*, 2008; Agaskar *et al.*, 2025].

These four map configurations (figure 1) cover a wide range from open to extremely congested environment, with differing start/goal distributions across trials to provide comprehensive evaluation conditions.

Robots and motion model. We set all robots to have a 0.05 radius (5cm in normalized units) and they move at a max speed of 1 cell per time step. Each robot can start and stop instantly (holonomic). Our experimental setup do not include any uncertainties or failures in motion execution in the planning stage.

Evaluation Metrics. We define the three metrics used throughout the paper for comparing planners. They are namely: a) Success Rate: We mark a trial as successful if we find trajectories for all robots without any collision. We report success rate as a percentage of trials which were successful out of total number of random trials (randomly generated start and goal positions); b) Planning Time: We measure the total time from providing the start/goal set to obtaining a solution for a trial; and c) Path Length: We use average of path length across all robots in a trial as a basic cost metric for a given trial.

XGBoost Training. We trained the XGBoost model on 8,362 samples generated from 450 planning scenarios (150 trials per map across three structured maps).

Scalability and Congestion. Our experiments are setup to demonstrate whether our methodology works in highly congested environments or not. Therefore, number of robots and map types together are chosen to demonstrate various levels of bottleneck congestion in path planning. At $n = 20$ robots, average conflicts over 10k trials when planning independently are: empty = 5, highways = 99, conveyor = 93, and drop-region = 87 conflicts.

4.2 Results

We evaluate our proposed SIPP-PP-LNS planner using the maps described in section 4.1 with up to 20 robots. We run 50 random trials i.e. randomized start and goal locations per map per robot count (4 maps * 6 robot counts * 50 trials = 1200). We compare our SIPP-PP-LNS planner against well established coupled and decoupled multi-robot motion planners (specifically, CBS, A*-ECBS and Prioritized Planning (PP)). We also show the usefulness of XGBoost-guided LNS repair component in SIPP-PP-LNS over our single stage planner without LNS, i.e. SIPP-PP. We provide additional context on benefits of our method in real-time settings by actively discussing recent results in the literature for a leading diffusion-based planner, Multi-robot Multi-model planning Diffusion (MMD) [Shaoul *et al.*, 2025].

Success Rate (figure 2): In the empty environment, all methods achieve near-perfect success even as the number of robots increase. As expected, decoupled methods perform well in this environment. However, we observe significant differences in performance for more structured and congested environment. In the Highways, Conveyor and Drop

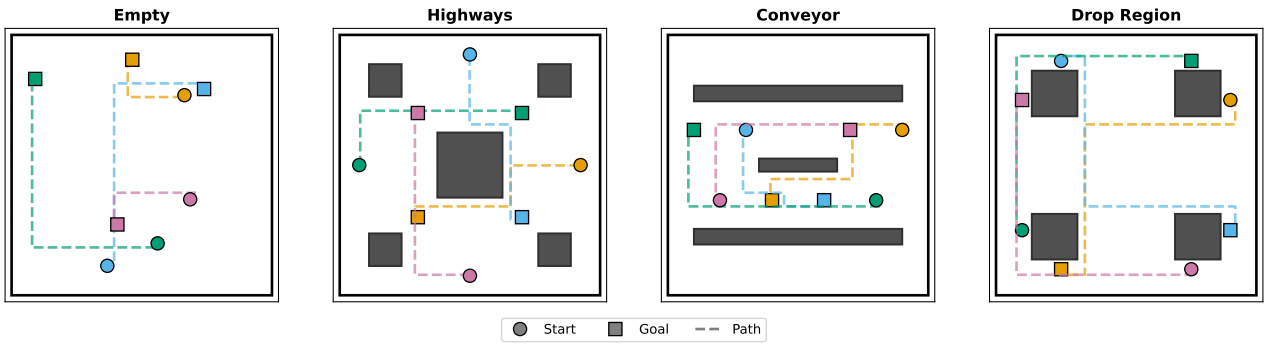

Figure 1: Planning maps with four robots containing static obstacles and structured regions, showing start and goal locations with corresponding collision-free trajectories in time.

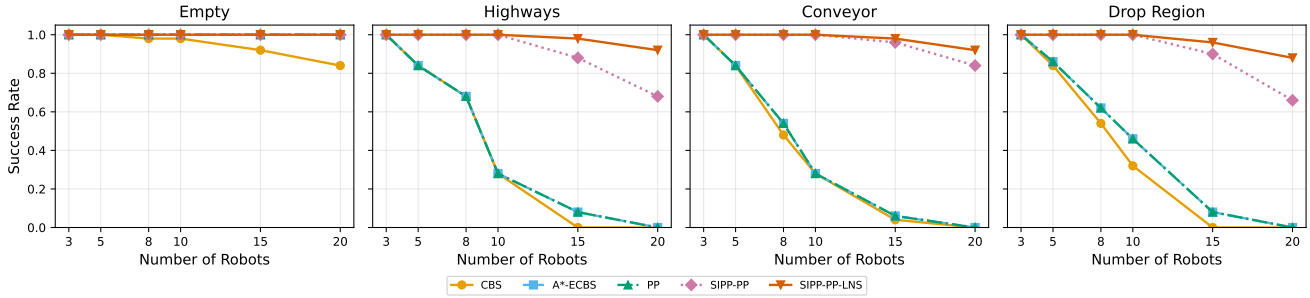

Figure 2: Planning success rate as a function of the number of robots across four environments of increasing structural complexity (Empty, Highways, Conveyor, and Drop Region). CBS, A*-ECBS and basic prioritized methods experience a rapid drop in success rate as robot density increases across all structured maps. In contrast, SIPP-PP-LNS maintains high success rates across all maps and scales robustly with number of robots.

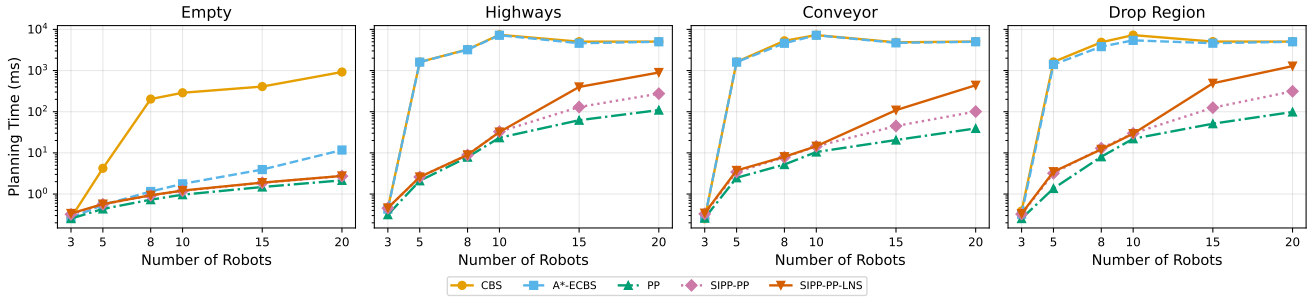

Figure 3: Average planning time (log scale) versus number of robots for each environment. CBS, A*-ECBS exhibit rapidly increasing planning time as the number of robots increase, generally coinciding with failures at larger team sizes. Prioritized methods are fast but less robust, while SIPP-PP-LNS achieves a favorable balance, maintaining bounded planning times while scaling to larger teams in structured environments.

region maps, success rates for CBS, A*-ECBS, and PP degrade quickly as the number of robots increases beyond 8–10. In contrast, while the SIPP-PP planner provides better success rates over PP, it's performance drops in high congestion/constrained settings. SIPP-PP-LNS consistently showed near-perfect success rate across all maps and robot counts, along with substantial improvement over SIPP-PP, highlighting its benefits in resolving complex spatial and temporal conflicts.

Planning Time (figure 3): In terms of planning time, SIPP-PP-LNS achieves a favorable balance, requiring only sub-second planning times up to 20 robots even in dense environments. This is particularly important, as even though Prioritized Planning can provide comparable planning time, the success rate of SIPP-PP-LNS is significantly better than PP as discussed in the previous subsection. Furthermore, recent diffusion-based learned approaches such as Multi-robot Multi-model planning Diffusion (MMD) have reported acceptable performance up to 20 robots, but they require multiple seconds per scenario (on the order of 10-15 seconds for

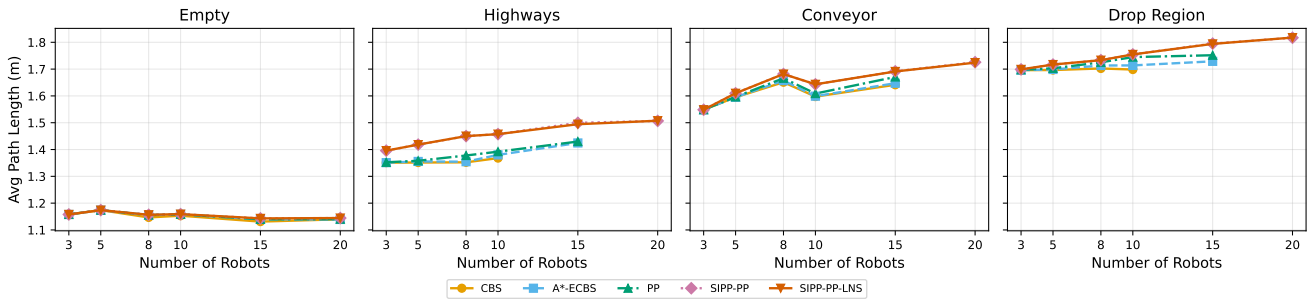

Figure 4: Average path length for successful runs as a function of the number of robots. For the empty map/unconstrained environments, all methods produce comparable average path lengths. For structured environments, SIPP-PP-LNS generates slightly longer trajectories than CBS, A*-ECBS, PP when they succeed, reflecting the trade-off between solution optimality and scalability.

k	Method	Success	Iters	Path Len
3	Random	62.1%	6.9	80.8
	Heuristic	62.0%	5.7	77.2
	XGBoost	**63.7%**	5.2	**75.2**
5	Random	60.0%	4.8	79.9
	Heuristic	56.7%	4.3	78.0
	XGBoost	**61.3%**	4.3	**77.6**

Table 1: LNS robot selection strategy comparison on 300 held-out conflict scenarios (20 robots). Success rate and average cumulative path length (m) after up to 15 LNS iterations.

small teams up to 9 robots). Our approach provides fast planning times compared to learning based planners.

Path Quality (figure 4): Lastly, we analyzed the average path lengths for successful runs. In unconstrained layouts (empty map), distribution of path lengths is similar across methods. However, in structured environments SIPP-PP-LNS and SIPP-PP often generate slightly longer paths than CBS, PP and A*-ECBS. SIPP-PP-LNS is much faster, produces higher success rates at large number of robots, but is not optimal. Similarly, compared to diffusion-based planners, such as MMD, SIPP-PP-LNS do not produce smooth, human-like motion trajectories learnt from data. This, however, is not concerning as usually in real-time industry deployments, the robotic floors are structured for rectilinear motion. Therefore, this distinction is critical: industrial deployments prioritize low latency and high re-planning throughput to support rapid response to task updates, whereas diffusion-based planners are more suited to scenarios where computation time is less of a concern and quality of motion is the primary objective.

4.3 Ablation Study: LNS Robot Selection Strategies

To study the effectiveness of learned selection in LNS repair phase, we did an ablation study to compare random sampling (randomly select k robots) and heuristic/count-based sampling (select k robots with highest conflict count) against a trained XGBoost model. We used 300 held-out conflict scenarios (100 per map for Highways, Conveyor and DropRegion) with 20 robots. We isolated the LNS phase by generating scenarios with guaranteed conflicts. We then applied LNS repair with $k \in \{3, 5\}$ robots selected per iteration and measured success rate (all conflicts resolved within 15 iterations) and average total path length for successful runs. Table 1 presents the results and shows learned strategy prioritizing high-impact robots in LNS, while additionally optimizing for path quality.

5 Conclusion, Limitations & Future Directions

In this paper, we presented a comprehensive solution for multi-robot motion planning and demonstrated that enhancing classical search algorithms and combining with statistical learning can provide competitive solutions for multi-robot planning, with fast inference required for practical deployment. Our approach provides a strong baseline and integration opportunity into real-world robotic fleet management systems where planning speed is of importance. Our method consistently provides feasible and collision-free trajectories for large robot teams in the millisecond to a few seconds regime. Fast planning time demonstrated by our methodology outperforms advanced planners by a large margin with excellent success rates in our method.

Few directions emerge from the limitations of this work: a) firstly, future work can focus on integrating a continuous optimization post-process to smooth trajectories and bring the quality of trajectories closer to diffusion-based planners, while maintaining performance; b) secondly, we can explore parallel planning, instead of sequential planning in our method, for robots that are distantly far or operate only in a specific sub-section of the entire map, as generally exhibited in real-world fulfillment centers with specialized areas of operation like picking, packing, sorting etc.; and, c) lastly, while we presented a lightweight XGBoost model use in our algorithm, as complexity grows, learning policies could take on a larger role in orchestrating multi-robot interactions in an intelligent way, including learning policies to choose among actions such as “replan the two most con- flicted robots” or “shift the schedule of a group of robots by a few time steps.”

In summary, our work demonstrated a fast MRMP algorithm for real-world deployment and provided a useful counterpoint to purely learning-based methods.

References

[Agaskar *et al.*, 2025] Ameya Agaskar, Sriram Siva, William Pickering, Kyle O'Brien, Charles Kekeh, Ang Li, Brianna Gallo Sarker, Alicia Chua, Mayur Nemade, Charun Thattai, Jiaming Di, Isaac Iyengar, Ramya Dharoor, Dino Kirouani, Jimmy Erskine, Tamir Hegazy, Scott Niekum, Usman A. Khan, Federico Pecora, and Joseph W. Durham. Deepfleet: Multi-agent foundation models for mobile robots, 2025.

[Bui, 2023] Hoang-Dung Bui. A survey of multi-robot motion planning. arXiv:2310.08599, 2023.

[Čáp *et al.*, 2015] Michal Čáp, Peter Novák, Alexander Kleiner, and Martin Selecký. Prioritized planning algorithms for trajectory coordination of multiple mobile robots. *IEEE Transactions on Automation Science and Engineering*, 12(3):835–849, 2015.

[Carvalho *et al.*,] Joao Carvalho, An T. Le, Mark Baierl, Dorothea Koert, and Jan Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 6466–6473.

[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 785–794. ACM, 2016.

[Cohen *et al.*, 2016] Liron Cohen, Tansel Uras, T. K. Satish Kumar, Hong Xu, Nora Ayanian, and Sven Koenig. Improved solvers for bounded-suboptimal multi-agent path finding. In *Proc. 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3067–3074, 2016.

[Erdmann and Lozano-Pérez, 1987] Michael Erdmann and Tomas Lozano-Pérez. On multiple moving objects. *Algorithmica*, 2:477–521, 1987.

[Hopcroft and Wilfong, 1986] John E. Hopcroft and Gordon T. Wilfong. Reducing multiple object motion planning to graph searching. *SIAM Journal on Computing*, 15(3):768–785, 1986.

[LaValle, 2006] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[Li *et al.*, 2021] Jiaoyang Li, Zhe Chen, Daniel Harabor, Peter J. Stuckey, and Sven Koenig. Anytime multi-agent path finding via large neighborhood search. In *Proc. 30th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4127–4135, 2021.

[Okumura, 2024] Keisuke Okumura. Engineering lacam*: Towards real-time, large-scale, and near-optimal multi-agent pathfinding. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1501–1509, 2024.

[Phillips and Likhachev, 2011] Michael Phillips and Maxim Likhachev. SIPP: Safe interval path planning for dynamic environments. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 5628–5635, 2011.

[Schwartz and Sharir, 1983] Jacob T. Schwartz and Micha Sharir. On the piano mover's problem: I. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, 36(3):345–398, 1983.

[Shaoul *et al.*, 2025] Yorai Shaoul, Itamar Mishani, Shivam Vats, Jiaoyang Li, and Maxim Likhachev. Multi-robot motion planning with diffusion models. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2025.

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

[Shaw, 1998] Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming (CP)*, volume 1520 of *LNCS*, pages 417–431. Springer, 1998.

[Stern *et al.*, 2019] Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, and T. K. Satish Kumar. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Symposium on Combinatorial Search (SoCS)*, pages 151–159, 2019.

[van den Berg and Overmars, 2005] Jur P. van den Berg and Mark H. Overmars. Prioritized motion planning for multiple robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 430–435, 2005.

[Wurman *et al.*, 2008] Peter R. Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, 29(1):9–20, 2008.

[Yu and LaValle, 2013] Jingjin Yu and Steven M. LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27, pages 1443–1449, 2013.

[Yu and LaValle, 2016] Jingjin Yu and Steven M. LaValle. Optimal multi-robot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics*, 32(5):1163–1177, 2016.