IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

K-ARC: Adaptive Robot Coordination for Multi-Robot Kinodynamic Planning

Mike Qin[1], Irving Solis[1], James D. Motes[1], Marco Morales[1,2], and Nancy M. Amato[1]

***Abstract*—This work presents Kinodynamic Adaptive Robot Coordination (K-ARC), a novel algorithm for multi-robot kinodynamic planning. Our experimental results show the capability of K-ARC to plan for up to 32 planar mobile robots, while achieving up to an order of magnitude of speed-up compared to previous methods in simulated scenarios. K-ARC is able to achieve this due to its two main properties. First, K-ARC constructs its solution iteratively by planning in segments, where initial kinodynamic paths are found through optimization-based approaches and the inter-robot conflicts are resolved through both optimization and sampling-based approaches. The interleaving use of both approaches allows K-ARC to leverage the strengths of each in different sections of the planning process where one is more suited than the other, while previous methods tend to emphasize one over the other. Second, K-ARC builds on a previously proposed multi-robot motion planning framework, Adaptive Robot Coordination (ARC), and inherits its strength of focusing on coordination between robots only when needed, saving computational effort. We show how the combination of these two properties allows K-ARC to achieve better overall performance in our simulated experiments with increasing numbers of robots, increasing degrees of problem difficulties, and increasing complexities of robot dynamics.**

***Index Terms*—Multi-robot Systems, Nonholonomic Motion Planning, Path Planning for Multiple Mobile Robots or Agents**

I. Introduction

MULTI-ROBOT systems have gained significant usage in real-world applications including warehouse automation, delivery, and surveillance tasks. Thus, it is important to develop scalable and efficient multi-robot kinodynamic planning (MRKP) algorithms that can compute trajectories for a team of robots that are collision-free and satisfy the robots' dynamic constraints.

MRKP is an instance of the more general problem of multi-robot planning, which can typically be approached in three ways. First, coupled methods treat the team of robots as one single entity, where the states of all the robots are combined and the problem can then be solved by any single-robot planner. Coupled methods are able to produce plans where high levels of coordination between the robots are needed but can often fail as the number of robots increases, due to the exponential growth of the search space, which makes finding a feasible solution very computationally expensive. Second, decoupled methods plan for robots individually and separately, where the plan for each robot is generated sequentially. Decoupled methods are typically faster in simpler problems but can fail when high levels of coordination between the robots are needed. This is because decoupled methods cannot adjust paths that are already found, leading to situations where robots may encounter deadlocks that cannot be resolved by only considering one robot's path at a time. To address MRKP, a naive coupled or decoupled adaptation of Kinodynamic Rapidly Exploring Random Trees (RRT) [1] can be used.

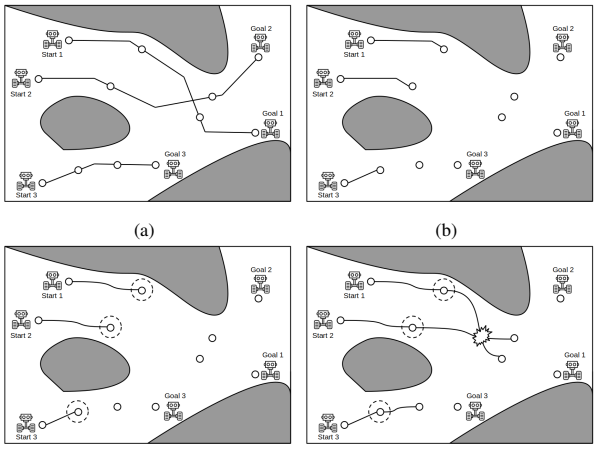

Fig. 1: K-ARC constructs the solution as follows: (a) each robot builds its own roadmap and finds an initial kinematic path, dividing it into segments with an equal number of timesteps; (b) the process begins with the first segment; (c) the kinematic path is then converted into a kinodynamically feasible trajectory; and (d) when processing the second segment, a conflict arises between robot 1 and robot 2, which is resolved using a multi-robot planner.

Hybrid methods combine the strengths of coupled and decoupled planning by managing inter-robot conflicts at a high level while planning individual robot paths separately. Most existing work follows this structure and builds on Conflict-Based Search (CBS) [2], a well-known algorithm for multi-agent pathfinding. These approaches differ in the specific planners used for multi-robot kinodynamic planning (MRKP), but share the same two-level structure: a high-level planner resolves conflicts between robots, and a low-level planner computes individual trajectories [3], [4], [5], [6].

Our previous work, Adaptive Robot Coordination (ARC) [7] can also be classified as a hybrid algorithm. ARC instead solves the multi-robot motion planning (MRMP) problem, a simplified version of MRKP that does not consider robot dynamics. The core idea of ARC is to dynamically adjust the dimensions of the problems, including the environment size and the number of robots involved, based on the levels of

Manuscript received: February 18, 2025; Revised June 13, 2025; Accepted July 14, 2025.

This paper was recommended for publication by Editor Lucia Pallottino upon evaluation of the Associate Editor and Reviewers' comments.

[1] Mike Qin, Irving Solis, James D. Motes, Marco Morales, and Nancy M. Amato are with the Siebel School of Computing and Data Science, University of Illinois Urbana-Champaign, Urbana, IL, 61801, USA. {yudiqin2, juanis, jmotes2, moralesa, namato}@illinois.edu.

[2]Marco Morales is also with the Instituto Tecnológico Autónomo de México, Department of Computer Science, México City, 01080, México.

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

coordination needed between the robots, making it an efficient algorithm in many different scenarios. In our original work, we only assumed holonomic robots and did not consider dynamic constraints on the robots, which puts restrictions on how the robots are allowed to move. As we will discuss in Section II-D, our original work is insufficient when this assumption is relaxed.

We present Kinodynamic Adaptive Robot Coordination (K-ARC), an efficient, hybrid multi-robot kinodynamic planner. In K-ARC, we find the initial individual path for each robot without taking into account inter-robot collisions and dynamic constraints. Instead of resolving inter-robot collisions iteratively on the entire paths, we operate on segments of the paths. We first divide the paths into an equal number of segments. In each segment, we find individual kinodynamically-feasible paths through optimization-based approaches. These paths are then checked for conflicts which are then resolved through both optimization and sampling-based approaches, using each approach where it is best suited, which we show significantly reduces computation time.

In summary, our contributions are:

- A novel MRKP algorithm, K-ARC, that adapts a ARC-styled subproblem-based multi-robot conflict resolution to consider robot dynamics.
- A novel framework that employs an integrated planning strategy that leverages the strengths of both sampling-based and optimization-based planning at different levels of problem complexity and size.
- An experimental analysis that shows K-ARC's ability to plan for up to 32 robots while achieving up to two orders of magnitude of speed-up across different scenarios compared to baseline methods from recent literature.

II. Preliminaries and Related Work

In this section, we introduce preliminaries, related work, and formally define the multi-robot kinodynamic planning (MRKP) problem. Additionally, we formally introduce our previous work on multi-robot motion planning, ARC, (which is extended in this manuscript) and discuss why it cannot be directly applied to solve multi-robot kinodynamic planning.

A. Motion Planning

Motion planning is the problem of finding a valid continuous path for a robot from a start to a goal configuration within the configuration space ($\mathcal{C}_{space}$) [8], the set of all possible robot configurations. A configuration $c \in \mathcal{C}_{space}$ encodes the robot's degrees-of-freedom (DOF), which fully parameterizes the robots' geometric pose. As the number of DOFs increases, representing the $\mathcal{C}_{space}$ explicitly becomes intractable [9]. In response, sampling-based methods were developed, which seek to approximate the $\mathcal{C}_{space}$ using a graph [10] (i.e., a roadmap) or a tree [11], rather than explicitly representing it.

B. Kinodynamic Planning

Kinodynamic planning takes the problem of motion planning and accounts for dynamic constraints on the movements of the robots, where the differential constraint $x' = f(x, u)$ dictates how the robots are allowed to move. Here, x is the state variable that encompasses the configuration ($c \subseteq x$) and its time derivatives, and u is the control variable.

While kinodynamic planning can be considered as a subproblem of motion planning, we use the term motion planning to strictly refer to the problem of kinematic motion planning, where robot dynamics are not taken into account, and to distinguish it from kinodynamic planning in this work.

Kinodynamic planning can be solved directly through sampling-based methods. Existing work consists of different variants of RRT [1][12][13][14], either with or without optimality guarantees. Alternatively, it can be solved through optimization-based methods [15] [16] [17]. Instead of explicitly planning in the $\mathcal{C}_{space}$, optimization-based approaches model the problem as a constrained optimization problem, where the obstacles and robot dynamics are defined as part of the cost function or added as constraints to the optimizer.

C. Multi-Robot Kinodynamic Planning (MRKP)

In this subsection, we first formally define the multi-robot kinodynamic planning (MRKP) problem and then discuss existing approaches.

1) Problem Definition: Given a team of $n \in \mathbb{N}$ homogeneous robots $R = \{r_1, r_2, ..., r_n\}$ operating in a shared workspace $\mathcal{W} \subseteq \mathbb{R}^d$ where $d \in \{2, 3\}$, our goal is to find dynamically feasible, collision-free trajectories for each robot from their respective start state to a goal region. Each robot r_i has state $x_i \in X_i$ and control input $u_i \in U_i$ and its motion is governed by the following continuous-time dynamics:

$$\dot{x}_i = f(x_i, u_i)$$

The dynamics are discretized over time with a fixed timestep Δt. During each timestep, the control input is assumed to be constant, and the state transition for each robot at timestep k is given by the following Euler integration:

$$x_{i,k+1} = x_{i,t} + f(x_{i,k}, u_{i,k})\Delta t$$

More formally, our problem is formulated as:

Find a trajectory $\{x_{i,1}, x_{i,2}, \ldots, x_{i,T}\}$ and control inputs $\{u_{i,1}, u_{i,2}, \ldots, u_{i,T-1}\}$ for each robot $r_i \in R$

subject to:

$$
\begin{aligned}
& x_{i,1} = x_{i,s} && (1)\\
& x_{i,T} \in X_{i,g} = \{x \in X \mid \|x - x_g\| \leq \alpha\} && (2)\\
& x_{i,k+1} = x_{i,k} + f(x_{i,t}, u_{i,t})\Delta t \quad \forall k \in \{1, 2, ..., T-1\} && (3)\\
& u_{i,k} \in U && (4)\\
& c_{i,k} \in \mathcal{W}_{\text{free}} \text{ where } c_{i,k} \subseteq x_{i,k} && (5)\\
& \|c_{i,k} - c_{j,k}\| \geq d_{\text{min}}, \quad \forall i \neq j && (6)
\end{aligned}
\tag{1}
$$

In this formulation, (1-2) are the start and goal constraints, and (3) corresponds to the dynamic constraints. In (4), we can only choose the control inputs from a set of allowable control inputs. (5) constrains the configuration of a robot's state to be within free space. (6) requires the configurations of any two robots to be at least a distance d_{min} away at any timestep.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

2) Existing Approaches: In existing methods for the MRKP problem, we see the use of sampling-based, optimization-based, or mixed methods like in single robot kinodynamic planning, and the use of decoupled, coupled, and hybrid strategies used in (holonomic) multi-robot motion planning (MRMP). Coupled methods typically apply a single robot method (like Kinodynamic RRT [1]) directly to the joint space. This offers coordination and (probabilistic) completeness guarantees at the cost of expensive planning times due to the complexity of the joint planning space. Thus most methods employ decoupled or hybrid strategies.

Decoupled methods (both sampling and optimization-based usually employ a priority-based approach, planning for robots sequentially, treating higher priority robots as dynamic obstacles [18], [19]. Priority-based approaches, while often quick, have been found to struggle in highly constrained environments with many robot-robot interactions. The approach in [4] attempts to alleviate this by leveraging hybrid planning of discrete paths, but still ultimately employs decoupled priority-based optimization with its coordination limitations.

In response, many recent hybrid methods have been developed, often extending CBS [2] by utilizing an individual kinodynamic planner and adapting the constraints to fit the planning space. We see a sampling-based version of this in K-CBS [6] and an optimization based version in CB-MPC [20]. A more recent CBS extension [3] leverages both sampling and optimization by adapting a discontinuity-bounded single-robot planner [21] to the multi-robot case, using trajectory optimization to smooth out infeasible connections.

All of these approaches have shown promise in solving MRKP problem, but are ultimately limited in either their scalability to large numbers of robot or their coordination capacity (usually due to decoupled prioritized planning). Our approach, K-ARC, instead uses them locally, first employing a pair of prioritized planners (optimization, then sampling-based) to try to solve the problem quickly before falling back to a more expensive Composite Kinodynamic RRT only when more coordination is required. We compare against two of the more recent CBS extensions in our experiments [6], [20].

D. Adaptive Robot Coordination (ARC)

ARC [7] is a MRMP method which initially plans decoupled, independent robot paths and then iteratively finds and resolves conflicts by creating local subproblems around the conflict. It then repairs the paths by replacing the conflicting portion of the paths with the subproblem solution. Subproblems initially limit the planners to the conflicting robots and a small region of the environment. Both the region and the number of robots are dynamically expanded based on local planner feedback.

Limitations for MRKP: The assumption of holonomic robots in ARC enables this insertion repair strategy, however, in MRKP, aligning the end of a local subproblem solution with the tail of the original kinodynamically feasible path is a much more challenging problem and may result in the need to replan the entire remainder of the path after the conflict (which might be thrown away again if there are additional conflicts with this path). Additionally, the small regions used for local subproblems in ARC are often too restrictive for resolving conflicts when considering dynamics constraints.

To tackle these limitations, we propose to construct solutions iteratively, instead of making modifications directly on the complete solution (as in ARC), and only limit subproblems in the number of robots, allowing use of the entire environment for conflict resolution. We explain our approach in detail in the following section.

III. Method

In this section, we give an overview of the high-level structure of our algorithm followed by a detailed description of its different planning stages.

A. Overview

Our algorithm (Algorithm 1) starts by finding the initial paths for each robot (line 3) using any kinematic sampling-based method. These paths will not used as the final solutions but rather as guidance for the kinodynamic solutions.

The individual paths are then divided into m equally sized segments for each robot where we obtain a goal configuration for each segment (line 6-9). For example, for a path consisting of a sequence of configurations for robot r_i, $\{c_{i,1}, c_{i,2}, \ldots, c_{i,T}\}$, the length of each path segment will be $\lceil \frac{T}{m} \rceil$ and the set of goals will be $G_i = \{x_{i,\lceil \frac{T}{m} \rceil}, x_{i,2\lceil \frac{T}{m} \rceil}, ..., x_{i,T}\}$ where each state in the set consists of its corresponding configuration. In addition, we also obtain and keep track of the initial kinematic path segment (line 10).

For each segment iteration, we define a single-robot kinodynamic planning problem which starts at the last reached state and targets a region centered around the current segment goal. This setup ensures continuity across segments. We then compute the kinodynamic trajectory using a single-robot planner, guided by the corresponding kinematic path segment.

The trajectory segments are then checked for inter-robot conflicts. As in our previous work [7], conflicts are resolved by defining a multi-robot kinodynamic planning (MRKP) problem. Each robot's query matches its current segment, and we use a set of MRKP solvers of varying complexity to handle conflicts requiring different coordination levels.

Once a subproblem is solved and trajectories are updated, the framework performs a global conflict check across all robot trajectories to ensure no new or residual conflicts remain. If a new conflict arises involving a different segment or additional robots, we integrate those into a new conflict resolution subproblem, as before.

After solving a segment, each robot's endpoint becomes the start for the next. Solutions are sequentially constructed until the final segment reaches the goal within tolerance.

B. Single robot trajectory Optimizer

The initial path constructed in each segment can be used as a reference for the trajectory optimizer to find the kinodynamic path. Any formulation which ensures the paths are dynamically feasible is viable within this framework. In an effort to minimize the total time of the resulting trajectory,

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Algorithm 1 Kinodynamic Adaptive Robot Coordination (K-ARC)

Input: A kinodynamic planning problem with an environment $\mathcal{E}$, a set of robots $\mathcal{R}$, a set of queries $\mathcal{Q}$.
Output: A set of valid, kinodynamically feasible paths $\mathcal{P}_k$.

1: $\mathcal{P} \leftarrow \emptyset$
2: **for** each robot $r_i \in \mathcal{R}$ **do**
3: $p_i \leftarrow$ `MotionPlanning`$(\mathcal{E}, \{r_i\}, \{q_i\})$
4: $\mathcal{P} \leftarrow \mathcal{P} \cup \{p_i\}$
5: **end for**
6: $m \leftarrow$ `number of path segments`
7: **for** each robot $r_i \in \mathcal{R}$ **do**
8: $p_{r_i} \leftarrow \mathcal{P}[i]$ ▷ retrieve the kinematic path for robot i
9: $G_{r_i} = \{g_1, g_2, \ldots, g_m\}$
10: $\mathcal{P}_{r_i} = \{p_1, p_2, \ldots, p_m\} \leftarrow$ `SegmentPath`(p_{r_i}, m)
11: **end for**
12: $\mathcal{P}_k \leftarrow \emptyset$
13: **for** $j \in m$ **do**
14: $\mathcal{P}_j \leftarrow \emptyset$
15: **for** each robot $r_i \in \mathcal{R}$ **do**
16: $q_{local} \leftarrow (g_{last}, \mathsf{g}_j)$ where $\mathsf{g}_j \in G_{r_i}$[j]
17: $\tau_{ij} \leftarrow$ `KinodynamicPlanning`$(\mathcal{E}, \{r_i\}, \{q_{local}\}, \mathcal{P}_{r_i}[j])$
18: $\mathcal{P}_j \leftarrow \mathcal{P}_j \cup \{\tau_{ij}\}$
19: **end for**
20: $\mathcal{C}$ = `FindConflict`$(\mathcal{P}_j)$
21: **while** $\mathcal{C} \neq \emptyset$ **do**
22: $\mathcal{E}', \mathcal{R}', \mathcal{Q}'$ = `CreateSubProblem`$(\mathcal{C}, \mathcal{P}_j, \mathcal{E})$
23: $\mathcal{P}'_j$ = `SolveSubProblem`$(\mathcal{E}', \mathcal{R}', \mathcal{Q}')$
24: **if** $\mathcal{P}'_j \neq \emptyset$ **then** ▷ conflict resolved
25: `UpdateSolution`$(\mathcal{P}_k, \mathcal{P}'_j)$
26: `UpdateLastGoals`$(\mathcal{P}'_j)$
27: $\mathcal{C}_{global}$ = `FindConflict`$(\mathcal{P}_k)$
28: **if** $\mathcal{C}_{global} \neq \emptyset$ **then**
29: $\mathcal{E}'', \mathcal{R}'', \mathcal{Q}''$ = `MergeConflict`$(\mathcal{C}_{global}, \mathcal{P}_k, \mathcal{E})$
30: $\mathcal{P}''$ = `SolveSubProblem`$(\mathcal{E}'', \mathcal{R}'', \mathcal{Q}'')$
31: `UpdateSolution`$(\mathcal{P}_k, \mathcal{P}'')$
32: **end if**
33: **else** ▷ if conflict not resolved
34: **return** $\emptyset$ ▷ return empty solution
35: **end if**
36: $\mathcal{C}$ = `FindConflict`$(\mathcal{P}_j)$
37: **end while**
38: `UpdateSolution`$(\mathcal{P}_k, \mathcal{P}_j)$
39: `UpdateLastGoals`$(\mathcal{P}_j)$
40: **end for**
41: **return** $\mathcal{P}_k$

we utilize the optimization formulation from [3], decoupling it for individual robots along their individual segments. In this, the cost function for a given timestep k is set to $J(x_{i,k}, u_{i,k}) = \beta_1 \|\mathbf{u}_{i,k}\|^2 + \Delta t$ to minimize the control input and total trajectory length ΔT with β_1 for regularization. For more details on this formulation, please see [3].

$$
\begin{aligned}
\underset{X,U,\Delta t}{\text{minimize}} \quad & \sum_{k=s\lceil \frac{T}{m} \rceil}^{(s+1)\lceil \frac{T}{m} \rceil} J(x_{i,k}, u_{i,k}) \\
\text{subject to} \quad & x_{i,s\lceil \frac{T}{m} \rceil} = x_{i,s} \quad (1) \\
& x_{i,(s+1)\lceil \frac{T}{m} \rceil} \in X_{i,g} \quad (2) \\
& x_{i,k+1} = x_{i,k} + f(x_{i,k}, u_{i,k})\Delta t \quad (3) \\
& u_{i,k} \in U \quad (4) \\
& c_{i,k} \in W_{free} \text{ where } c_{i,k} \subseteq x_{i,k} \quad (5)
\end{aligned} \tag{2}
$$

We also note that this optimizer is applied iteratively to each segment s during the planning process (Algorithm 1, lines 15-19) as opposed to the entire path at once, hence the start and goal equal to $s\lceil \frac{T}{m} \rceil$ and $(s+1)\lceil \frac{T}{m} \rceil$ in the cost function and lines (1) and (2) respectively. Doing so results in smaller, easier optimization problems. The algorithm also has the option to fall back to Kinodynamic RRT in the case that the trajectory optimization exceeds some time threshold to maintain probabilistic completeness (though in practice, this never occurred in our experiments).

C. Subproblem Construction and Planning

The kinodynamic paths obtained from the trajectory optimizer are checked for conflicts. A conflict is defined as a tuple of $(x_{i,k}, x_{j,k})$ for robots r_i and r_j and a timestep k. Given such a conflict, we define a local subproblem $(\mathcal{R}', \mathcal{Q}')$ around the conflict (Algorithm 2) where $\mathcal{R}' = r_i \cup r_j$ merges the involved robots. The local query $\mathcal{Q}'$ defines the local start and goal state for each robot and is obtained by taking the start state and goal state for the segment where the conflict occurs along with some tolerance for reaching the goal state.

The conflict resolution follows a similar process as in ARC [7], as shown in Algorithm 2, although several modifications are made due to the added complexity on the robots' movement. First, planning is not limited to a local environment, as smaller environments overly restrict the planning space for conflict resolution, and is instead allowed to operate in the entire environment. Second, if a solution is not found $(\mathcal{E}, \mathcal{R}', \mathcal{Q}')$, we adapt the subproblem by setting the start query to the robot's previous segment start and the goal query to its next segment goal (line 10), as opposed to in ARC where the queries are obtained by small incremental expansions. These changes address the added complexity of kinodynamic planning, where larger state spaces enable a higher chance of finding a feasible trajectory, especially for robots with highly constrained motions. Lastly, in subproblem planning (line 4), the following hierarchy of solvers are used:

- Prioritized trajectory optimization
- Decoupled Kinodynamic RRT
- Composite Kinodynamic RRT

The hierarchy follows the intuition of using faster methods first, increasing robustness as problem complexity grows. We start with prioritized trajectory optimization: for robots $r_1, r_2, ..., r_k$, we find kinematic paths through a group planner. These are then sequentially optimized—r_1's path goes into Equation (2), r_2's is optimized with a constraint to maintain distance from r_1, and so on. This method resolves mild conflicts but terminates if the solver exceeds the iteration limit without a solution.

If prioritized optimization fails, we move to sampling-based solvers: first Decoupled Kinodynamic RRT, then Composite Kinodynamic RRT. If needed, the Composite Kinodynamic RRT jointly explores all involved robots' configuration space and stops when a node or sample budget is reached. These solvers are based on the methods discussed in Section II-C2 with implementation variations discussed in Section IV-A. This hierarchy balances efficiency and robustness—in practice, most conflicts are resolved by the first solver.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Algorithm 2 SolveSubProblem

Input: A subproblem with environment $\mathcal{E}$, a set of robots $\mathcal{R}$, a set of queries $\mathcal{Q}'$. A set of MRKP solvers $\mathcal{S}$.
Output: A set of valid local paths $\mathcal{P}'$.

1: $\mathcal{P}' \leftarrow \emptyset$
2: **while** $\mathcal{P}' == \emptyset$ **do**
3: **for** s **in** $\mathcal{S}$ **do**
4: $\mathcal{P}' \leftarrow$`SolveMRKP`$(s, \mathcal{E}', \mathcal{R}', \mathcal{Q}')$
5: **if** $\mathcal{P}' \neq \emptyset$ **then** ▷ subproblem solved
6: **return** $\mathcal{P}'$
7: **end if**
8: **end for**
9: **if** $\mathcal{E}' \neq \mathcal{E}$ **or** $\mathcal{Q}' \neq \mathcal{Q}$ **then**
10: `AdaptSubProblem`$(\mathcal{E}', \mathcal{R}', \mathcal{Q}')$
11: **else**
12: **return** $\emptyset$ ▷ if subproblem not solved,
13: **end if** ▷ return empty local solution
14: **end while**

D. Design choices

We recognize that there are several design decision made in our instantiation of this framework. Here we discuss those decisions and their alternatives.

1) *Initial pathfinding:* In Algorithm 1, line 3, we plan paths for the robots individually, without checking for conflicts. The purpose of this is to quickly find feasible paths that can be used as rough guidance. Alternatively, we can choose to run ARC [7] where the resulting kinematic paths will be conflict free, hopefully reducing the number of kinodynamic conflicts. In some preliminary experiments evaluating this alternative design, we found that naively solving kinematic conflicts with ARC often created difficult guiding paths for the kinodynamic planner and did not offer a significant reduction in the number of conflicts found during the main planning stage resulting in equivalent or even slightly worse overall planning times. This does rule out the potential of effectively exploiting kinematic conflict resolution to reduce the computational burden at the kinodynamic stage, but the investigation of how to do so is outside the scope of this work.

2) *Path segmentation:* In Algorithm 1, lines 9-10, we split each robot's path into m equal-length segments (segments may differ across robots if their total path lengths vary). This simplifies tracking the milestones robots achieve as we loop through the algorithm. Alternatively, we could fix the segment length, which may yield more segments for longer paths. This can help synchronize robot movements, as they must reach segment milestones simultaneously. While beneficial for paths of varying lengths, we must manually insert waiting states for early-arriving robots (creating overhead), or synchronization is lost and the benefit disappears.

E. Theoretical Properties

K-ARC maintains the same completeness properties as ARC [7]. When the algorithm fails to solve a subproblem corresponding to a set of path segments, it expands this subproblem to include the path segments immediately before and after the current subproblem. In the worst case, all robots are involved, and this expansion continues until the subproblem is equivalent to the global problem. At this point, the completeness of K-ARC is equivalent to the most complete method in the hierarchy of solvers, or a different, last resort solver, can be specified to maintain completeness.

K-ARC also does not provide any kind of optimality guarantees just like ARC [7]. The greedy nature of solving local subproblems limits considerations of global path alternatives. Additionally, the kinodynamic tracking of the initially proposed kinematic paths further restricts any kind of optimality guarantee for the final kinodynamic solution.

IV. Experiments

In our experiments, we compare our method against two hybrid MRKP methods: K-CBS [6], which uses sampling-based replanning, and CB-MPC [5], which uses optimization-based planning. These methods were chosen because, like our approach, they coordinate robots through hybrid strategies, and they have also demonstrated effectiveness in planning for many robots. We test all methods on three virtual scenarios and discuss the results, highlighting the benefits of our dual-method approach that combines sampling and optimization.

A. Experiment Setup

We implement two CBS-based baselines: CB-MPC [5] and K-CBS [6] (using vanilla CBS search methods [2]). Our low-level planner for CB-MPC, uses MPC with RRT generated reference paths. Constraints are encoded as signed distance requirements (e.g., for constraint $(c, [t, t+h])$, the robot must stay at least one radius from configuration c during that interval). Our trajectory optimizers for both K-ARC and CB-MPC are implemented in CasADi [22] and solved with IPOPT [23]. For simplicity, both obstacle and robot geometries are modeled as polyhedrons, and collision constraints use shortest distance computations.

The low-level planner for K-CBS and the Decoupled Kinodynamic RRT in the K-ARC solver hierarchy both use a simplified kinodynamic RRT [1] which finds an independent feasible trajectory, then lazily applies Safe Interval Path Planning (SIPP) [24] constraints to handle conflicts, modifying the path and inserting waiting where needed. After an empirically derived conflict threshold is reached, new paths are sampled. In the relatively simple scenarios considered, we found this approach effective and faster than handling conflict constraints directly within kinodynamic RRT.

The number of segments m used in K-ARC is empirically derived for each scenario, choosing it large enough to ensure that transitions remain smooth and tractable. We scale it with the number of robots and the environment size to maintain consistency across scenarios.

All the methods are implemented in C++ in the open-source Parasol Planning Library, and the experiments are run on a machine with 32-core Intel(R) Core(TM) i9-14900K and 64 GB of RAM. Each method is given 20 trials for each scenario, with a timeout of 600 seconds.

B. Scenarios

1) Open Cross: We examine a simple cross scenario as illustrated by Fig. 2a. Robots on the same row need to swap

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

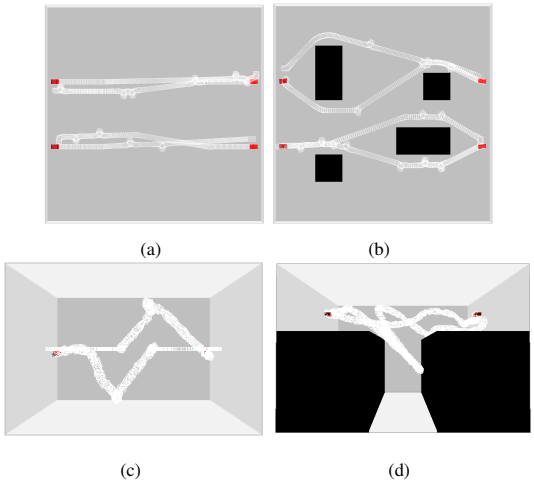

Fig. 2: The experiment environments. The paths shown in white are produced by K-ARC. (a) 4-robot open cross scenario (b) 4-robot cluttered cross scenario (c) 2-robot quadrotor scenario (d) 2-robot quadrotor inlet scenario

positions in an empty environment. This is a simple scenario to evaluate how many robots the methods are able to scale up to when conflicts are likely. In this, and the following cluttered cross environment, we used both first-order and second-order unicycle robot models.

2) Cluttered Cross: We examine a harder scenario where the environment is filled with cluttered obstacles (Fig. 2b). The robots have the same start and goal positions as in open cross. The obstacles add extra constraints to the robots movements, making this a more difficult problem, and are used to test how the methods are able to handle complex environments.

3) Quadrotor Cross: We examine a scenario where multiple 6-DOF quadrotor robots move across an empty environment (Fig. 2c). This scenario tests the methods' ability to handle robots with complex constraints and high-dimensional state space. We also relax the goal tolerance to allow handling larger numbers of robots.

4) Quadrotor Inlet: We examine a highly-constrained scenario as illustrated by Fig. 2d where 2 quadrotor robots need to switch positions in an environment with narrow passages and an inlet to allow extra space for the robots to move. The tight obstacle constraints, combined with the complex dynamic constraints of the quadrotor robots, make this scenario particularly challenging, where the methods can only handle 2 robots.

C. Robot Models

We briefly describe the robot models used.

a) First-Order Unicycle: This model assumes the robot can directly control its linear velocity v and angular velocity ω. The state consists of its position and orientation, $[x, y, \theta]$.

b) Second-Order Unicycle: This model includes dynamics for acceleration. The state is $[x, y, \theta, v, \omega]$, where v and ω are the linear and angular velocities, and the controls a and α are the linear and angular accelerations.

c) Second-Order Quadrotor: We use a second-order quadrotor model as a simplified version of the actual quadrotor dynamics. The state includes position $[x, y, z]$ and orientation $[\phi, \theta, \psi]$ (roll, pitch, yaw), as well as the linear velocity $[\dot{x}, \dot{y}, \dot{z}]$ and angular velocity $[\dot{\phi}, \dot{\theta}, \dot{\psi}]$. The controls are thrust T and torque $[\tau_\phi, \tau_\theta, \tau_\psi]$.

D. Results

We evaluate the methods based mainly on the runtime. The runtime as shown in the tables and figures is the amount of time in seconds a method spends in finding a solution. We also evaluate the methods on path cost as a secondary metric. We calculate the path cost as the sum of timesteps of all trajectories times the time resolution. Additionally, we show the number of conflicts each method has to resolve in each of the scenarios.

1) Open Cross: In this scenario to test the scalability of the methods, we find that K-ARC achieves significantly better runtime across all team sizes and is the only method able to scale up to 32 robots, as shown in Fig. 3. We attribute this performance to K-ARC's strategy of resolving conflicts through localized subproblems, which involve only the robots and time intervals directly affected by the conflict. This contrasts with CBS-based methods like K-CBS and CBS-MPC, which replan entire trajectories for individual robots from scratch during each conflict resolution. The ability of K-ARC to isolate conflicts and resolve them independently allows it to reduce unnecessary computation and coordinate more efficiently at scale.

Comparing K-ARC with CB-MPC, which also uses optimization, we find that K-ARC consistently achieves better runtime. Although CB-MPC applies a model predictive control strategy with a sliding time window, it still requires full-trajectory optimization in response to each new constraint. In contrast, K-ARC only solves shorter, targeted subproblems per conflict, which significantly reduces the overhead associated with replanning and contributes to its superior scalability.

For CBS variants, CB-MPC has slightly better performance than K-CBS for the first-order robot, in both runtime and path cost. While K-CBS can achieve better path cost for the second-order robot, it cannot keep up after 16 robots in both cases. This is mainly due to the large number of constraints imposed on the CBS conflict tree. This is less of an issue for CB-MPC, mainly because the constraints imposed on the optimizer are generally not too tight with the large open space.

K-CBS resolves significantly more conflicts than K-ARC and CB-MPC but with comparable runtime. This stems from how conflicts are handled: K-ARC and CB-MPC use trajectory optimizers in continuous space, offering more options at higher computational cost. In contrast, K-CBS uses SIPP on a discrete graph, which limits options but simplifies the problem and enables faster resolution, as shown by its shorter average query times.

2) Cluttered Cross: In this scenario to test how the methods handle complex environments, we find that no methods can scale over 8 robots and have overall poorer performance compared to the open cross scenario, as shown in Fig. 4. This is expected as this is a highly constrained scenario. However, K-ARC is still able to outperform both baseline methods in the 8-robot scenarios, whereas K-CBS has better performance in the 2-robot and 4-robot scenario.

CB-MPC is slower than both K-CBS and K-ARC. K-ARC also performs worse here than in the open-cross scenario. This is due to the added complexity of obstacle constraints, which

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

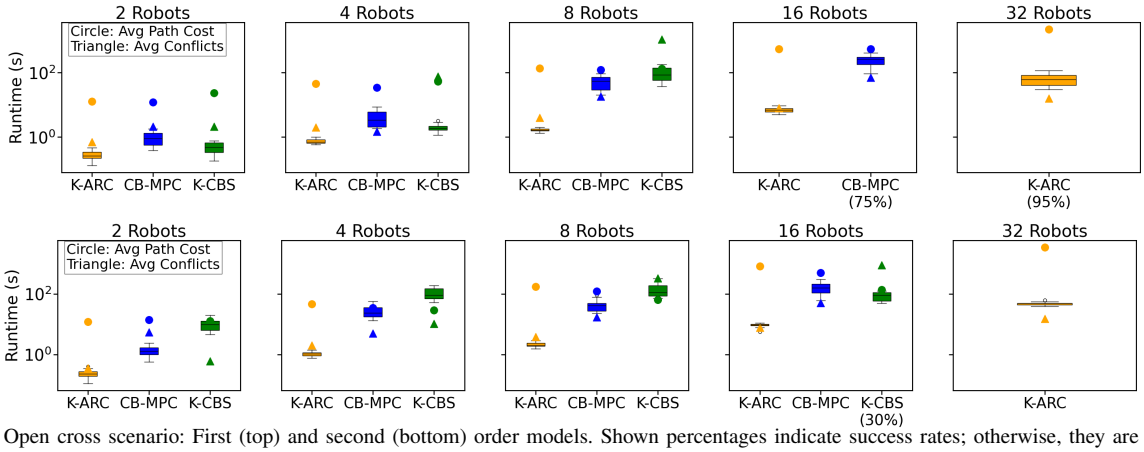

Fig. 3: Open cross scenario: First (top) and second (bottom) order models. Shown percentages indicate success rates; otherwise, they are 100.%

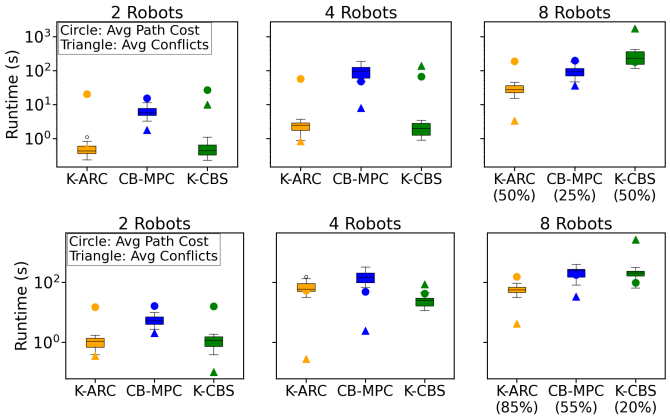

Fig. 4: Cluttered Scenario: First (top) and second (bottom) order models.

affects optimization-based and sampling-based approaches differently. In trajectory optimization, obstacle constraints apply at every state, creating tight restrictions. In contrast, sampling-based methods sample only collision-free states, avoiding the need for checking during graph queries.

3) Quadrotor Cross: In this scenario to test how the methods handle complex robot dynamics, we find that only K-ARC and K-CBS are capable of planning up to 16 robots (Fig. 5), while CB-MPC has trouble planning for 8 robots. In the 16-robot scenario, K-ARC not only has less than half of the runtime compared to K-CBS, it also has less than half of the path cost. We recognize that this discrepancy is due to the randomness in K-CBS's purely sampling-based approach. Given the large search space of this scenario, K-CBS tends to produce poor-quality tree in the initial pathfinding stage, which impacts both its runtime and path cost. CB-MPC, on the other hand, consistently achieves the best path cost. However, its optimization-based process soon becomes computationally expensive given the large state space of the quadrotor robot model and as the number of robots increases.

4) Quadrotor Inlet: In this highly constrained scenario, only K-ARC and K-CBS successfully solve the problem, as shown in Table I. The faster runtimes and improved success rate of K-ARC demonstrate the advantage of being able to employ the higher coordination potential of coupled methods when appropriate.

Robots	Model	Method	Runtime		Path Cost		Conflicts		Success
			Avg	Std Dev	Avg	Std Dev	Avg	Std Dev	
2	2nd order	**K-ARC**	**69.34**	81.98	168.5	40.37	0.56	0.50	100%
		CB-MPC	-	-	-	-	-	-	0%
		K-CBS	144.71	116.66	221.22	84.07	2.28	1.34	90%

TABLE I: Results for the quadrotor inlet scenario

E. Experimental Insights into Local Conflict Resolution

To better understand K-ARC's behavior during conflict resolution, we analyzed solver usage and subproblem expansions in a controlled experiment with 100 successful trials of the Open-Cross scenario with 8 robots. We observed (Table II) that most conflicts were resolved using prioritized trajectory optimization, with fewer requiring Composite Kinodynamic RRT and only a small portion needing Decoupled Kinodynamic RRT. Subproblem expansion was rarely necessary.

These results highlight K-ARC's flexibility to scale conflict resolution complexity when necessary. There are many design decisions which impact the distribution of solver utilization and success. For example, we set a relatively low conflict threshold for Decoupled Kinodynamic RRT (as discussed in Section IV-A), so it has a low success rate here. Instead, we more quickly defer to Composite Kinodynamic RRT.

The high number of conflicts resolved by prioritized optimization aligns with the segment size used, which is not too short. Larger segments offer enough flexibility for simple solvers to resolve conflicts. Likewise, expansion was rarely needed. Still, the ability to expand remains important in tighter scenarios. For example, when two robots must cross a narrow corridor in opposite directions, expansion allows one to exit and let the other pass. We note that most subproblems (though not all) are solvable without requiring an expansion. This is in contrast to ARC which utilized frequent expansions, a consequence of allowing the solver to explore the entire environment instead of constraining plans to small local regions.

V. Conclusions

In this paper, we presented Kinodynamic Adaptive Robot Coordination (K-ARC), a multi-robot kinodynamic planning algorithm that extends a previously proposed framework. K-ARC integrates sampling-based and optimization-based techniques at different planning stages, leveraging the strengths of

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

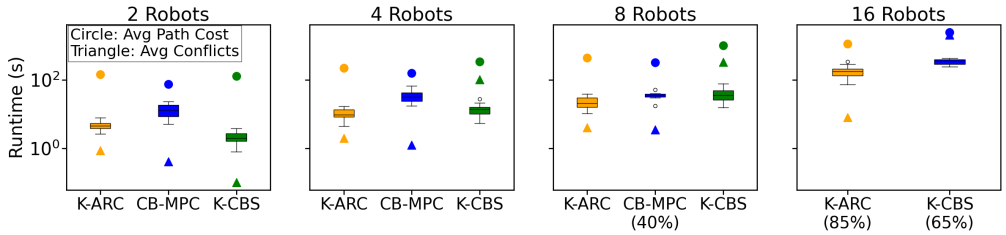

Fig. 5: Results of the quadrotor cross scenario.

Total	Prioritized Trajectory Optimization	Decoupled Kinodynamic RRT	Composite Kinodynamic RRT
252 (100 %)	141 (55.9 %)	16 (6.3 %)	95 (37.6 %)

Total	No expansion	One expansion
252 (100 %)	245 (97.2 %)	7 (2.8 %)

TABLE II: Distribution of local solvers and subproblem expansions in 100 successful trials.

both. Experimental results show it outperforms baseline methods across various scenarios, demonstrating the effectiveness of combining discrete sampling with trajectory optimization in multi-robot planning.

While our current experiments focus on static environments, K-ARC's modular structure, especially its segment-wise construction and local conflict resolution, makes it a strong candidate for extension to dynamic environments with moving obstacles. Its incremental trajectory construction also enables online planning, where segments can be updated as new information arrives. Additionally, the flexible hierarchy of local solvers in K-ARC allows integration of real-time methods such as time-elastic bands or potential fields to help avoid collisions during execution. In future work, we will explore these directions to broaden K-ARC's applicability to real-time, dynamic multi-robot navigation.

Acknowledgement

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

References

[1] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 1, pp. 473–479 vol.1, 1999.

[2] 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.

[3] A. Moldagalieva, J. Ortiz-Haro, M. Toussaint, and W. Hönig, "db-cbs: Discontinuity-bounded conflict-based search for multi-robot kinodynamic motion planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14569–14575, 2024.

[4] J. Li, M. Ran, and L. Xie, "Efficient trajectory planning for multiple non-holonomic mobile robots via prioritized trajectory optimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 405–412, 2021.

[5] A. Tajbakhsh, L. T. Biegler, and A. M. Johnson, "Conflict-based model predictive control for scalable multi-robot motion planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14562–14568, 2024.

[6] J. Kottinger, S. Almagor, and M. Lahijanian, "Conflict-based search for multi-robot motion planning with kinodynamic constraints," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13494–13499, 2022.

[7] I. Solis, J. Motes, M. Qin, M. Morales, and N. M. Amato, "Adaptive robot coordination: A subproblem-based approach for hybrid multi-robot motion planning," *IEEE Robotics and Automation Letters*, vol. 9, no. 8, pp. 7238–7245, 2024.

[8] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications ACM*, vol. 22, pp. 560–570, October 1979.

[9] J. F. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press, 1988.

[10] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[11] S. LAVALLE, "Rapidly-exploring random trees : a new tool for path planning," *Research Report 9811*, 1998.

[12] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE Conference on Decision and Control (CDC)*, pp. 7681–7687, 2010.

[13] D. J. Webb and J. van den Berg, "Kinodynamic rrt*: Optimal motion planning for systems with linear differential constraints," 2012.

[14] K. Hauser and Y. Zhou, "Asymptotically optimal planning by feasible kinodynamic planning in a state–cost space," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1431–1443, 2016.

[15] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE International Conference on Robotics and Automation*, pp. 4569–4574, 2011.

[16] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*, pp. 489–494, 2009.

[17] J. Schulman, Y. Duan, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, pp. 1251 – 1270, 2014.

[18] B. Cain, M. Kalaitzakis, and N. Vitzilaios, "Mk-rrt*: Multi-robot kinodynamic rrt trajectory planning," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 868–876, 2021.

[19] J. Chen, J. Li, C. Fan, and B. Williams, "Scalable and safe multi-agent motion planning with nonlinear dynamics and bounded disturbances," in *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 11237–11245, 2021.

[20] A. Tajbakhsh, L. T. Biegler, and M. Johnson-Roberson, "Conflict-based model predictive control for scalable multi-robot motion planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14562–14568, 2024.

[21] W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, 2018.

[22] J. Andersson, J. Gillis, G. Horn, J. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, 07 2018.

[23] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization," *Comput. Chem. Eng.*, vol. 33, pp. 575–582, 2009.

[24] 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, 2011.