A Benchmark for Optimal Multi-Modal Multi-Robot Multi-Goal Path Planning with Given Robot Assignment

Valentin N. Hartmann*, Tirza Heinle*, Stelian Coros

***Abstract*— In many industrial robotics applications, multiple robots are working in a shared workspace to complete a set of tasks as quickly as possible. Such settings can be treated as multi-modal multi-robot multi-goal path planning problems, where each robot has to reach *an ordered sequence* of goals. Existing approaches to this type of problem solve this using prioritization or assume synchronous completion of tasks, and are thus neither optimal nor complete. We formalize this problem as a single path planning problem and introduce a benchmark encompassing a diverse range of problem instances including scenarios with various robots, planning horizons, and collaborative tasks such as handovers.**

Along with the benchmark, we adapt an RRT* and a PRM* planner to serve as a baseline for the planning problems. Both planners work in the composite space of all robots and introduce the required changes to work in our setting.

Unlike existing approaches, our planner and formulation is not restricted to discretized 2D workspaces, supports a changing environment, and works for heterogeneous robot teams over multiple modes with different constraints, and multiple goals.

Videos and code for the benchmark and the planners is available at https://vhartman.github.io/mrmg-planning/.

I. Introduction

As adoption of robots increases and simple tasks become more and more automated, it will be more and more important to deploy solutions that are not only able to work longer and cheaper but also are competitive in throughput with humans. In order to achieve this, in many cases, multiple robots need to be used and effectively coordinated in the same workspace: Enabling motion planning for multiple tasks with multiple robots is crucial in order to maximize the usefulness of robots in industrial settings. While workcells with multiple robots exist, the robots typically act independent from each other in order to simplify the programming and avoid dealing with robot-robot interactions.

Most work in continuous multi-robot planning is focusing on single-goal settings where all robots starting from their respective starting points at the same time and reach their respective goals simultaneously [1]–[3]. Conversely, in most real use cases, multiple robots need to do multiple tasks in sequence, e.g., welding multiple points in a welding application, or picking and placing multiple things after another in order to sort objects. Since the robots act in the same environment in these scenarios and thus possibly block each other from doing their respective task, we can not formulate the problem at hand as a *sequence of path planning problems*, but need to solve the multi-robot multi-goal planning problem if we want to find an optimal solution.

All authors are with the Computational Robotics Lab (CRL), ETH Zürich.
* indicates equal contribution.

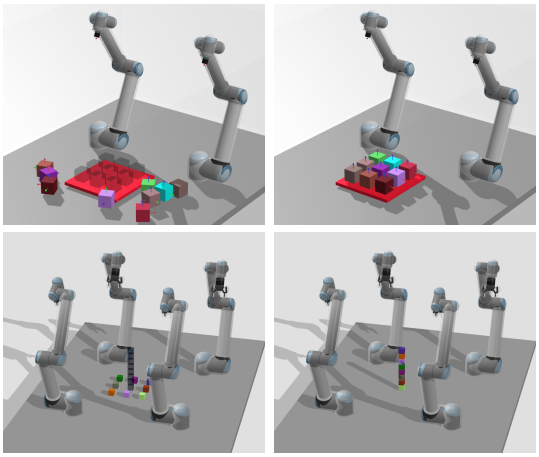


Fig. 1: Examples of problems with their initial state (left) and goal state (right): (top) two robots cooperating to reorient boxes, (bottom) four robots stack boxes on top of each other.

The work on multi-modal planning can be seen as multi-goal planning problem. Multi-modal path planning [4]–[7] finds paths through sequences of *modes*, i.e., through variations of a continuous configuration space that occur, e.g., by grasping an object, or moving an object in the workspace. This means that in each *mode* a different set of constraints is active. Most work on multi-modal path planning for robots only considers single-robot settings.

On the other hand, multi-agent path finding (MAPF) [8], [9] deals with high numbers of agents, but typically considers grid-like 2d environment with disk robots, as e.g., found in warehouses. MAPF was extended to consider multiple goals in the setting of multi agent pickup and delivery (MAPD) [10]. These approaches do not directly transfer to a changing environments and continuous configuration spaces.

Our contributions in this paper are

- a formalization of multi-modal, multi-robot, multi-goal path planning in continuous spaces,
- two probabilistically complete and almost-surely asymptotically optimal planners based on an RRT* planner, and a PRM* planner, respectively.

We open-source an easily accessible multi-modal, multi-robot, multi-goal motion planning benchmark containing 21 different base-scenarios with up to 74 subgoals, that can be further adjusted in difficulty by changing the number of robots and tasks.

Our goal with the planners is not to provide planners with

TABLE I: Comparison of path planning approaches.

Approach	Examples	Multi-Robot	Multi-Goal	Multi-Modal	Continuous	Complete	Opt.
MAPF	[8], [9]	✓				✓	✓
MAPD	[10], [11]	✓	✓			✓	✓
Multi-Robot Planning	[12]–[15]	✓			✓	✓	✓
Multi-Modal Planning	[4], [5]		✓	✓	✓	✓	✓
Prioritized planning	[16]–[18]	✓	✓	✓	✓		
Ours		✓	✓	✓	✓	✓	✓

the highest performance possible, but rather to to make the problem accessible and provide baselines.

II. Related Work

Most multi robot planning work focusing either on the setting where all robots share the same work and configuration space, or on solving single goal problems. It is commonly assumed that the environment stays completely static during planning, therefore excluding manipulation planning settings. We try to group the different research areas and their respective focus in Table I. In the following, we discuss single-goal multi-robot planning and MAPF more in-depth.

A. Continuous Multi Robot Planning

A simple approach to multi-robot-planning is planning in the composite space of all robots [14], and applying standard methods such as RRT(*) [19] or PRM(*) [20] in this higher dimensional space. This approach is not scaleable to many robots or high degrees of freedom.

There are various approaches to tackle this limitation. Discrete RRT (dRRT(*) [13], [21]), builds a roadmap for each robot, and combines them to implicitly form a tensor graph. Instead of planning naively on the implicit graph, an RRT-like strategy is proposed to explore this implicit graph. Fast-dRRT [17] applies this approach sequentially on the subproblems to solve multi-goal problems. M* [1] proposes to plan in the separate spaces when possible, and planning jointly only where required, i.e., when conflicts between the single-robot plans arise. Similarly, conflict based search (CBS) [22] plans for each robot individually, and introduces constraints to solve conflicts between single agent plans. CBS originates form the MAPF-setting, but can also be applied to continuous multi robot planning problems [12], [23], [24].

Prioritized planners as proposed in, e.g., [15], [16], [18], [25] achieve more scaleable solvers by not requiring completeness and optimality. Similarly, [26] proposes a scaleable approach that is based on operator splitting and sampling, and while the approach is complete, it is not optimal.

To the best of the authors knowledge, there are no multi-modal multi-robot multi-goal planners that are complete and optimal in the general setting. The prioritized planners can be used to plan for multi goal problems by incrementally generating a path per agent, and considering it as fixed for the later planning problems as done in [16], [18].

A slightly different approach is taken in [27], where plans for the separate robots are planned, and then - if conflicts between the paths arise - fixed by introducing pauses.

Compared to the previously introduced works, we are interested in optimally solving multi-robot multi-goal planning problems without any restricting assumptions. Further, we are interested in manipulation planning problems, i.e., problems where the environment changes through the actions of the robot.

B. Multi Agent Path Finding

Compared to multi robot path planning, multi agent path finding [9] generally refers to planning in grid-like 2D environments with disk-robots, as found in warehouses. A common approach to solve this problem is conflict based [22] or priority based search (PBS) [28]. There are many works that improve upon the initial approach, such as improved conflict based search [29], or bounded suboptimal CBS [30], which scale to 1000s of agents.

Multi agent pickup and delivery (MAPD) refers to the setting where we do not only intend to go from point A to point B, but traveling between multiple points of interest, as part of a single planning problem [10]. While MAPD can be framed as lifelong MAPF problem, i.e., once an agent has finished its task of the MAPF problem, a new goal is assigned to the robot, these solvers are suboptimal [31]. Multi-label A* [11] (MLA*) extends the classic A* algorithm to a sequence of multiple goals, and is able to optimally solve MAPD problems. Both MAPF and MAPD planners typically assume that the environment remains unchanged over the planning duration.

III. Multi Robot Multi Goal Path Planning

Informally, our objective is to find a path for each robot that passes through a sequence of goals that minimizes a cost (e.g., minimizing the latest completion time of all robots) while being collision free. Some of the goals might involve multiple robots: a handover of an object between two robots implies a constraint on two robots. A goal could imply a *mode transition*, i.e., a robot grasping something and thus changing the environment for the remaining planning problem.

A. Preliminaries

Before formalizing the problem, we introduce the components that are required: the tasks, the concept of a mode, and the state space that we plan in. We follow the work from Thomason [4], and generalize it to multiple robots. Compared to [4], we *do not* consider task planning.

1) Task: A task S consists of the robots that are assigned to the task, and a set of constraints g[1] that need to be fulfilled to consider the task done. A task can have post-conditions that can alter the scene-graph of the environment, i.e., which objects are linked to each other. As example, consider the task 'robot r_1 grasps object o_1': Here, the goal constraints

[1]Typically, the constraint g is a single goal pose or a goal region, but can also be a more complex constraint such as a grasp constraint.

are that the robot is grasping o_1, and the post-condition is that o_1 is linked to the end-effector of the robot.

We use $s \in \mathcal{S}_R = \mathcal{S}_{r_1} \times ... \times \mathcal{S}_{r_n}$ to denote the *task assignment* of all robots, where $\mathcal{S}_{r_i}$ is the set of tasks that are assigned to robot r_i.

2) Modes: The constraints from a task can be fulfilled by various different poses, which might affect the environment that we plan in. Consider again the grasping-task: the robot is able to fulfill a grasping constraint by grasping from any side. This changes the collision geometry that we need to consider in the rest of the planning problem, and it might influence how the constraints for follow-up tasks can be fulfilled. We use *mode* to refer to the combination of the discrete task assignment s (which implies a scene-graph), and the poses of all movable objects, defined by the relative transformation to their parent-frames. Compared to related work ([4], [32]), in this work a *mode* implies what each of the robots is doing.

3) Task sequence: A task sequence can be specified either as total ordering, fully determining which tasks come after another, or as dependency graph by specifying which tasks depend on others being fulfilled, which can be processed into a full ordering per robot. We require the terminal task to involve all robots in both the graph and the sequence. Both described specifications can be expressed as dependency graph. We thus describe a task sequence as a dependency graph $G = (S, E)$, with the tasks $S \in \mathcal{S}_{r_i}$, and the directed edges between tasks $E \in (\mathcal{S}_{r_i} \times \mathcal{S}_{r_j})$. Some examples of how a task sequence can be specified are in Fig. 2.

A given (partial) task sequence implies all possible transitions between task assignments (Fig. 3), and thus all valid task assignment sequences. We use $\mathcal{M}$ to denote the set of all valid task assignment sequences that bring us from the start to the goal, and obey the constraints imposed by the dependency graph.

4) State Space: The space in which we describe a path is the composite space of all robots, all objects, and which tasks are currently active per robot:

$$\mathcal{Q} = \underbrace{\mathcal{Q}_{r_1} \times \cdots \times \mathcal{Q}_{r_N}}_{\mathcal{Q}_R} \times \mathcal{Q}_o \times \underbrace{\mathcal{S}_{r_1} \times \cdots \times \mathcal{S}_{r_N}}_{\mathcal{S}_R}. \quad (1)$$

Here, $\mathcal{Q}_{r_i}$ is the configuration space of robot r_i, and correspondingly, $\mathcal{Q}_R$ is the composite configuration space of all robots; $\mathcal{Q}_o$ is the composite configuration space of all objects.We use $\mathcal{Q}_{\text{free}} \subseteq \mathcal{Q}$ to denote the part of the configuration space that is collision-free.

Clearly, not all degrees of freedom are actuated. We assume that we can only plan for the robots' degrees of freedom directly, and all others need to be influenced indirectly.

B. Problem formulation

Bringing everything together, a multi-modal, multi-robot, multi-goal path planning problem is given by the tuple $(R, \mathcal{Q}_R, q_{\text{start}}, G)$, where R is the set of robots, $\mathcal{Q}_R$ is the configuration space of the robots, q_{start} is the initial state, and G is the dependency graph of the tasks. In the following, we will use q^{r_i} for the pose of robot r_i.

We want to find a collision free path $\pi(t) : \mathbb{R} \to \mathcal{Q}_R$ and the task assignment sequence $s(t) : \mathbb{R} \to \mathcal{S}_R$, while minimizing some cost function $c(\cdot)$. The path π maps time to the joint robot state $\mathcal{Q}_R$, and the mode sequence $s(t)$ maps time to the task assignment $\mathcal{S}_R$, which together imply the scene-graph at a time, and thus the poses of all objects.

This can be written as optimization problem:

$$\min_{\pi,s} \quad c(\pi, s) \quad (2a)$$
$$\text{s.t.} \quad \pi(0) = q_{\text{start}} \quad (2b)$$
$$g(\pi(t), s(t) \leq 0) \quad \forall\, t \quad (2c)$$
$$s \in \mathcal{M} \quad (2d)$$
$$\pi(t) \in \mathcal{Q}_{\text{free}}(s(t)) \quad \forall\, t, \quad (2e)$$

where (2a) is the cost term, (2b) is the initial condition, (2c) and (2d) ensure that the constraints that are imposed by the mode sequence hold, and that the mode sequence is a valid one, and finally (2e) is the constraint for ensuring that the path is collision free.

1) Cost functions: We are often interested in finding the minimum makespan plan. However, purely minimizing makespan can lead to optimal paths that bring undesired side-effects, e.g., containing unnecessary movement. Thus, we consider cost functions of the family

$$c(q_1, q_2) = (1-w) \max_r ||q_1^r - q_2^r||_2 + w \sum\nolimits_r ||q_1^r - q_2^r||_2. \quad (3)$$

where if w is small, we get a minimum makespan optimization problem (which is 'regularized' by the path length), and if w is 1 we sum the cost of the robots.

IV. Planners

We introduce an RRT based planner and a PRM based planner as baselines for the problems we propose. In both these sampling based planners, the space we plan in is the previously defined one. However, instead of directly sampling the (unactuated) object poses, we sample $x = (q, m)$, where $q \in \mathcal{Q}_R$ is a pose containing all robots' poses and m is a mode that determines the task assignment and all object poses. We elaborate on how modes are sampled in Sections IV-A and IV-B. We then proceed in both the PRM* and the RRT* planners as their standard version would, with an adjusted distance function that accounts for the connectivity between the modes. Before discussing the details of the planners, we discuss the distance function used in the planners, and the concept of the informed set [33].

1) Distance functions: Similar to [4], we assume that any node in a different mode is inaccessible except through the transition nodes:

$$d(x_1, x_2) = \begin{cases} d_{\text{mode}}(q_1, q_2) & \text{if } m_1 = m_2, \\ \infty & \text{otherwise.} \end{cases}$$

For the in-mode distance any metric can be chosen; the work in [34] discusses distance metrics in multi-robot problems in more depth. For our planners, we choose the maximum of the per-robot L_2 distance, as this performed best in our experiments.

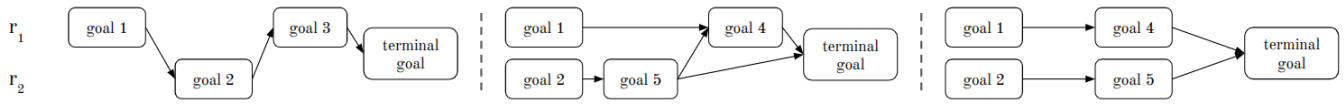


Fig. 2: Examples for the different ways in which a goal sequence can be specified: **(left)** As a sequence of goals **(middle)** as a partially ordered sequence and **(right)** as a per-robot ordered sequence.

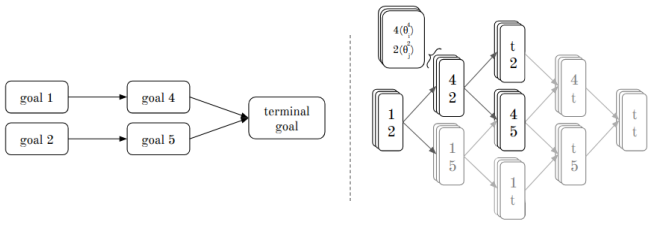


Fig. 3: Example of a mode-graph that is implied by the dependency graph on the left. The modes with full opacity are an example for how a set of modes can be incrementally be reached. Each mode can be reached with many different parameterizations θ_i, as indicated for mode [4, 2].

2) Locally informed sampling: The informed set [33] can be used to restrict the set of points that we need to consider for planning to the ones that can improve the current solution. The informed set in our problem is

$$\mathcal{Q}_{\text{inf}} = \{q \,|\, c_{\text{lb}}(q_{\text{start}}, q) + c_{\text{lb}}(q, Q_{\text{goal}}) \leq c_{\text{best}}\}, \qquad (4)$$

with Q_{goal} denoting the set of possible goal poses.

In our setting, the optimal paths are typically long compared to the size of the state space, and therefore, the informed set is typically larger than the valid configuration space (similar as in [35]).

Given that we do know that a solution needs to pass through some regions in space, the standard approach of taking the cost function as c_{lb} is not approximating the true path cost well. The main problem then is that the path constraints from the tasks are constraints for subspaces of the full composite space (i.e., even if we know that robot r_i needs to be at pose q_{g_1} first, robot r_j might be unconstrained), and it is thus not trivial to construct a good lower bound for the path-cost. Additionally, compared to the euclidean norm, our cost function does not easily allow for direct sampling from the informed set, therefore requiring rejection sampling, which can be compute intensive.

To alleviate these issues, we propose *locally informed sampling*: instead of using the start and the goal pose in Eq. (4), we randomly sample two points q_i, q_j on the path, and do informed sampling with these points:

$$\mathcal{Q}_{\text{lis}} = \{q \,|\, c(q_i, q) + c(q, q_j) \leq c^{\pi}(q_i, q_j)\}, \qquad (5)$$

where $c^{\pi}(\cdot)$ is the cost to reach a point on the path along the best found path.We extended this to the sampling of modes by sampling from the set of modes that are in-between the modes that q_i, q_j are in respectively. This set of in-between modes can be inferred from the task dependency graph.

Algorithm 1: Composite Multi Goal RRT*.

$T \leftarrow \{q_0\}$, $M \leftarrow \{m_0\}$
while `not terminated` **do**
  $q_{\text{rnd}}, m_{\text{rnd}} \leftarrow$`SampleWithGoalBias()`
  $n_{\text{close}} \leftarrow$`T.get_nearest(`$m_{rnd}$`,` q_{rnd}`)`
  $n_{\text{new}} \leftarrow$`Steer(`$n_{\text{close}}, q_{\text{rnd}}$`)`
  `T.add(`n_{new}`)`
  `T.rewire(`n_{new}`)`
  if n_{new} is subgoal
    $M \xleftarrow{+}$ `getNextMode(`n_{new}`)`
  if improved solution **and** do shortcutting
    `sol`$\leftarrow$`shortcut`
    `T.addPath(sol)`

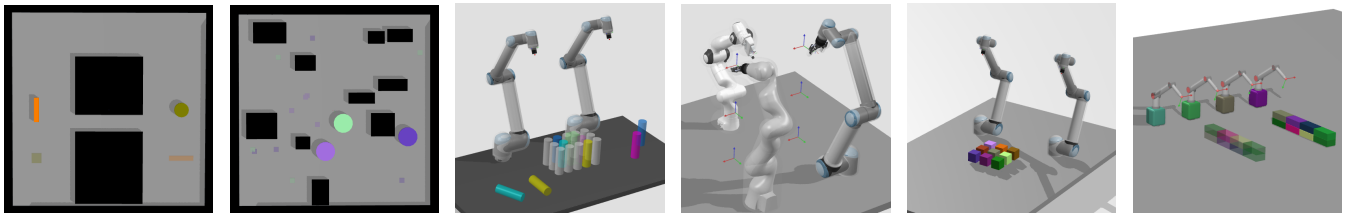
Algorithm 2: Composite Multi Goal PRM*.

$G \leftarrow \{q_0\}$, `sol`$\leftarrow \{\}$
while `not terminated` **do**
  $G \leftarrow$`SampleTransitionNodes()`
  $G \leftarrow$`SampleReachedModes()`
  if `GoalNodeFound()`
    `sol`$\leftarrow$`G.search()`
    if improved solution **and** do shortcutting
      `sol`$\leftarrow$`shortcut`
      `G.addPath(sol)`
return `sol`

A. Composite RRT*

We introduce the required changes to a standard RRT* planner [19], [36] to make it work in the multi-modal multi-goal setting (Algorithm 1): The core ideas are the same - we randomly sample a state consisting of a pose and a mode. We maintain a set of previously reached modes M, which is initialized with the start mode. We then sample a valid mode/pose combination either by sampling a mode uniformly at random from the reached modes M, and then sampling a pose uniformly in this mode, or if we previously found a path, we use locally informed sampling to sample the mode and the pose. We then search the closest state in our tree (according to the distance function), and grow the tree towards the randomly sampled state. The new state is added to the tree and the tree is rewired. If the new state is a transition (i.e., a pose fulfilling the goal constraints of a task), we add the mode that we reach from this transition to the set of reached modes. We do shortcutting (Section IV-C) once a solution is found to improve convergence of the planners [37].

We also introduce a bidirectional variant, which samples goals in each mode (which are used as starts in the next mode), and then runs bidirectional RRT* in each mode.



(a) 2D hallway. (b) 2D random. (c) Bottle insertion. (d) Multi waypoints. (e) Rearrangement. (f) Mobile assembly.

Fig. 4: The initial states of a selection of problems that are available in the benchmark. The poses that have to be reached by the robots or objects are drawn with lower opacity or indicated with a marker.

Algorithm 3: Multi-robot multi-modal partial shortcutting.

```
while not terminated do
    r ← sampleRobot()
    i, j ← sampleIndices()
    if IsShortcuttable(r, i, j)
        p ← constructShortcutPath(sol, r, i, j)
        if couldImproveCost(p) and
          isCollisionFree(p)
            sol ← updatePath(sol, p)
return sol
```

B. Composite PRM*

We introduce the required changes to a PRM-based planner [20] to adapt it to our setting (Algorithm 2). We build our roadmap by first sampling a batch of transition nodes (i.e., nodes that fulfill the constraints of a task), which enlarges the set of reached modes. We then sample a batch of nodes in the modes that we reached so far (using locally informed sampling if we found a solution, uniformly at random if not). If we found a transition that allows us to reach to goal, we then run an A* search over the multi-modal graph formed by all the samples so far (which is similar to MLA* [11]). We implement this search using an edge queue, which allows delaying edge validation until the edge is inspected, therefore minimizing collision checking effort. To select neighbors, we choose a radius based approach, with the radius determined as in [36]. Similar to the RRT planners, we do shortcutting after a path was found that improves the current solution.

Heuristics: For the search of the graph that we are incrementally building, we require a heuristic. The heuristics in our setting present a tradeoff between being useful and being fast to compute. We compute a heuristic by running a reverse search on the transition nodes to compute a lower bound on the cost to reach a mode. The heuristic is then the sum of the cost to reach a specific transition and the cost to reach the goal from this transition mode: $h(q) = \min_i c(q, q_i^{\text{trans}}) + c_{lb}(q_i^{\text{trans}})$, where q^{trans} are the poses belonging to transition nodes in a mode.

C. Postprocessing

We also introduce a path post-processing method that is specifically tailored to the multi-goal motion planning problem. Our approach (Algorithm 3) builds on top of partial shortcutting [38]. The main idea is that we do not shortcut all dimensions of the path at once, but select a subset of dimensions from the composite space, and try to shortcut only this subset, while keeping the rest of the plan the same. Instead of uniformly sampling from all possible subsets, we notice that in practice it is more efficient to sample a robot, and then shortcut all robot dimensions at once. This enables improving the unconstrained parts of the path (i.e., the parts that do not influence a mode-transition), while still keeping the simplicity of the shortcutting approach.

D. Proof sketches

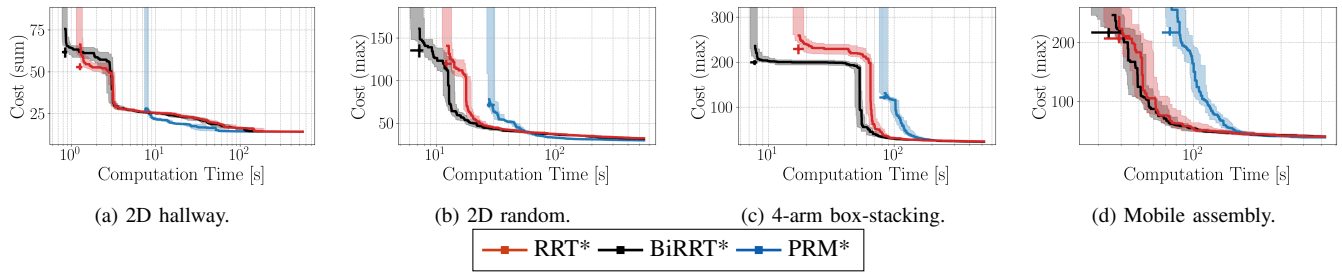
Both planners uniformly sample all transitions, and sample uniformly within the modes, therefore eventually discovering all possible modes. Within the modes, both planners work as their respective base version, and are therefore probabilistically complete, and will eventually connect to the transitions. If a path exists, the discovered transitions will lead to the terminal mode, which contains the goal pose. Thus, both planners will find a path if one exists, and are therefore probabilistically complete.

The proof sketch for almost-sure asymptotic optimality follows similar logic: The A* search that we run in the PRM-planner is optimal, and therefore, as the number of uniform-random samples approaches infinity, we converge to the optimal paths within the modes, and to the optimal mode-transitions. Similarly, the rewiring-step in the RRT-planners is the same as in RRT*, and thus inherits the optimality guarantees from RRT* even over multiple modes.

V. Benchmark & Experiments

The main contribution of this work is the open source benchmark implementing a variety of scenarios with a diverse sets of robots and tasks. The 21 base scenarios range from simple settings where the optimal solution path is known to help validate properties of the planner, to more complex scenarios with up to 5 robots (with a total of 30 degrees of freedom), and up to 22 goals. For many of the problems, there is both a version with a full task ordering, and one using a dependency graph as task specification. We show a selection of scenarios in Fig. 4.

As the main goal of this work is the introduction of the problem statement and making the problem accessible, and not the fastest possible planner, the benchmark and planners are implemented using Python, while the computationally expensive parts, i.e., collision checking and kinematics are



(a) 2D hallway. (b) 2D random. (c) 4-arm box-stacking. (d) Mobile assembly.

Fig. 5: Evolution of median cost over time along with the 95% non-parametric confidence intervals over 50 runs.

leveraging the python bindings of RAI[2]. This backend can however easily be replaced, allowing for, e.g., GPU parallelization if the backend supports it.

A. Experiments

We present a selection of tasks to showcase the different types of scenarios and task sequence specifications and compare the planners on them. We report results for four representative scenarios: a 2D scenario with two robots that is similar to the classic wall-gap, where the robots have to switch positions and go back again (6D configuration, 3 subgoals, Fig. 4a), a 2D scenario with 3 agents that have to reach a number of goals in sequence (9D conf. space, 13 subgoals, Fig. 4b), a scenario with 4 robot arms where 8 boxes have to be placed on top of each other (28D, 17 subgoals, Fig. 1), and a scenario involving 4 mobile manipulators rearranging a wall (24D, 17 subgoals, Fig. 4f). In these scenarios, the task sequences are fully determined for the two 2D scenarios, and *only partially given* for the robot-scenarios, i.e. the order and timing of each task needs to be figured out by the planner. The tasks are assigned randomly to a robot, and in all these instances, there is a single goal pose for each subgoal.

If not specified otherwise, we consider the cost function with $w = 0.01$, i.e., the min-makespan problem with a small path-length regularization, and we report the median solution costs over 50 runs along with confidence intervals.

The experiments were run with Python 3 on a Ryzen 7 5800X (8-core, 4'491 MHz) and 32 GB RAM.

B. Results

Figure 5 shows the resulting cost evolution plots. In general, it is expected that the solution times for these problems are higher than solution times for a 'standard' path planning problem. This is partially due to the length of the plans, and therefore the larger collision checking effort, in addition to the large search space, and partially due to the problem structure: If the planner finds a path to a better mode-transition, the rest of the path has to be completely replanned for all following modes and mode-transitions.

The RRT-based planners are up to an order of magnitude faster at finding an initial solution, while the PRM-planner tends to converge faster to the optimal solution once an initial path was found. Of the RRT-planners, the bidirectional version is usually faster.

[2]https://github.com/MarcToussaint/robotic

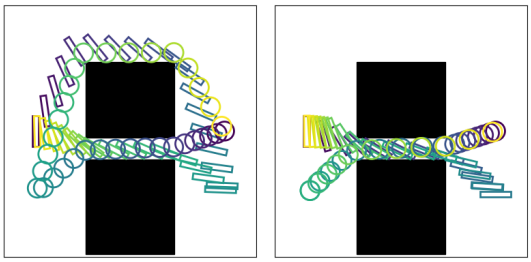


Fig. 6: The optimal paths when using a max **(left)** or sum **(right)** cost function in the 2D hallway scenario, where the robots have to reach a goal on the other side and return. Color indicates time from purple (start) to yellow (end).

Influence of the cost function: We illustrate the difference of the sum-cost and the max-cost on the hallway example in Fig. 6: Both robots go through the wall gap twice when using the sum-cost, and do not make use of the passage at the top, compared to the (regularized) max-cost, where they make use of the passage at the top.

C. Ablations

In the following, we present ablations to explain design decisions of the planning algorithms.

1) Locally informed sampling: As alluded to earlier, only using naive sampling in the problems we consider here is insufficient, as the informed set is large compared to the configuration space, and thus in many cases does not help much even compared to only using uniform sampling. In Fig. 7, we show the planners using locally informed sampling, and compare it to the naive informed sampling. The planners using locally informed sampling outperform the planners that do not. However, it can also be seen that in the higher dimensional environment, the optimum is not reached in the allotted planning time by either of the groups of planners.

2) Shortcutting in the planner: Shortcutting a path after it was found leads to faster convergence than using only informed sampling, as can be seen in Fig. 8. Shortcutting in the planner does not affect the optimality or completeness properties of the planners, but does significantly improve convergence by quickly cutting down on the maximum cost that the planners need to consider. This effect is more pronounced in high-dimensional scenarios.

3) Connection strategy: Using k-nearest neighbors in the PRM-planner results in finding the initial solution faster, but

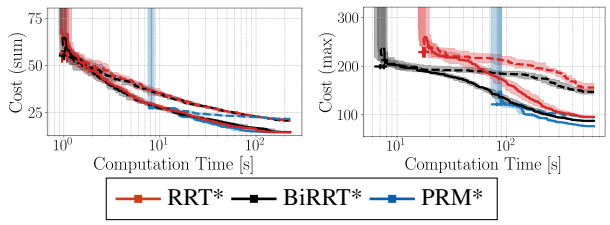


Fig. 7: Local (solid line) and global (dashed line) informed sampling on the 2D hallway **(left)** and the 4-arm box-stacking **(right)** scenarios.

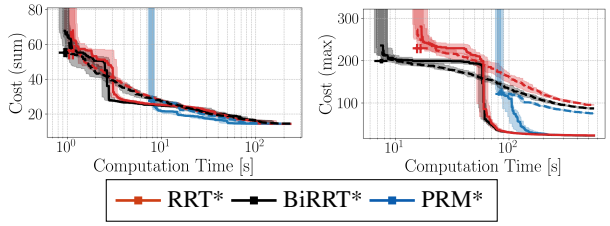


Fig. 8: Planners with (solid line) and without (dashed line) short-cutting on the 2D hallway **(left)** and the 4-arm box-stacking **(right)** scenarios. All planners are using locally informed sampling.

slower convergence to the optimal solution (see Fig. 9). We find that this is the case since the samples that we have are non-uniformly distributed around the transitions. The difference in speed can largely attributed to the fact that many more connections are added to the queue in the radius based connection strategy compared to the k-nearest approach. This leads to a higher branching factor, and thus a slower planner.

VI. Discussion & Limitations

There are some limitations on the scaling of the planners: that they do not scale well both with the number of robots (due to the increase in state-space dimensions) and they do not scale well with the number of goals. These limitations stem mostly from the decision to plan in the composite space and to treat the complete problem as one single big planning problem.

While we introduce a problem formulation that deals with dependency graphs, this increases the size decision space, and we believe that there could be a better way to approach the problem than a brute force search, which the sampling based planners do here.

Despite these limitations, we find that the planning in composite space is likely good enough for many complex industrial settings, where the number of used robots does not go much beyond 4 robots. We believe that a promising approach for improving the planner performance is leveraging the structure of the problem space, i.e., using the fact that we are dealing with multiple robots and multiple sub-goals, by, e.g., not planning over the full horizon directly, but only planning for a finite horizon.

VII. Conclusion

We presented a formalization of the multi-modal multi-robot multi-goal motion planning problem, and a benchmark

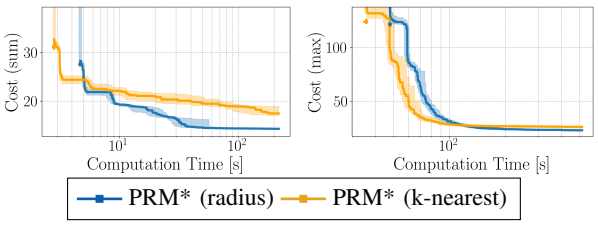


Fig. 9: Different connection strategies for the PRM planner on the 2D hallway **(left)** and the 4-arm box-stacking **(right)** scenario.

containing diverse problems reaching from simple environments with short goal sequences, to relatively long horizon problems requiring coordination of multiple robots. We introduced two planners that are probabilistically complete, with both of them being asymptotically optimal, based on an RRT planner, and a PRM planner respectively.

The planners that we propose are not expected to scale to very long horizon problems or problems with a large amount of robots. However, we believe that considering the problem in composite space helps understand the implication that, e.g., priorization in planning has in the continuous space, and through that better understanding, help develop better planners.

There are some clear improvements possible, such as such exploiting the multi-robot structure of the planning problems by adapting, e.g., dRRT to our problems. Further, the nature of the mode families [39], or the multi-query-like structure of the problem [40] could be used. We also believe that receding horizon-planners are an interesting avenue of research in order to transition this purely offline planning approach to real world execution.

Finally, the formulation we propose supports extension of the transition logic to multi-robot task and motion planning by changing how the next mode, respectively the mode sequence is generated.

Acknowledgments

The authors thank Miguel Zamora and Yijiang Huang for insightful discussions and comments on both text and illustrations.

References

[1] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial intelligence*, vol. 219, pp. 1–24, 2015.

[2] R. Shome and L. E. Kavraki, "Asymptotically optimal kinodynamic planning using bundles of edges," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 9988–9994.

[3] S. Lin, A. Liu, J. Wang, and X. Kong, "A review of path-planning approaches for multiple mobile robots," *Machines*, vol. 10, no. 9, p. 773, 2022.

[4] W. Thomason, M. P. Strub, and J. D. Gammell, "Task and Motion Informed Trees (TMIT*): Almost-surely asymptotically optimal integrated task and motion planning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 370–11 377, 2022.

[5] K. Hauser, "Task planning with continuous actions and nondeterministic motion planning queries," in *Proc. of AAAI Workshop on Bridging the Gap between Task and Motion Planning*, 2010.

[6] P. Englert, I. M. R. Fernández, R. K. Ramachandran, and G. S. Sukhatme, "Sampling-based motion planning on sequenced manifolds," in *Proc. of Robotics: Science and Systems (R:SS)*, 2021.

[7] P. S. Schmitt, W. Neubauer, W. Feiten, K. M. Wurm, G. V. Wichert, and W. Burgard, "Optimal, sampling-based manipulation planning," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3426–3432.

[8] J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1163–1177, 2016.

[9] R. Stern, "Multi-agent path finding–an overview," *Artificial Intelligence: 5th RAAI Summer School, Dolgoprudny, Russia, July 4–7, 2019, Tutorial Lectures*, pp. 96–115, 2019.

[10] H. Ma, J. Li, T. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding for online pickup and delivery tasks," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '17. International Foundation for Autonomous Agents and Multiagent Systems, 2017, p. 837–845.

[11] F. Grenouilleau, W.-J. Van Hoeve, and J. N. Hooker, "A multi-label a* algorithm for multi-agent pathfinding," in *International Conference on Automated Planning and Scheduling*, vol. 29, 2019, pp. 181–185.

[12] A. Moldagalieva, J. Ortiz-Haro, M. Toussaint, and W. Hönig, "db-cbs: Discontinuity-bounded conflict-based search for multi-robot kinodynamic motion planning," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 14 569–14 575.

[13] R. Shome, K. Solovey, A. Dobson, D. Halperin, and K. E. Bekris, "drrt*: Scalable and informed asymptotically-optimal multi-robot motion planning," *Autonomous Robots*, vol. 44, no. 3-4, pp. 443–467, 2020.

[14] G. Sanchez and J.-C. Latombe, "Using a prm planner to compare centralized and decoupled planning for multi-robot systems," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol. 2, 2002, pp. 2112–2119 vol.2.

[15] A. Orthey, S. Akbar, and M. Toussaint, "Multilevel motion planning: A fiber bundle formulation," *The international journal of robotics research*, vol. 43, no. 1, pp. 3–33, 2024.

[16] V. N. Hartmann, A. Orthey, D. Driess, O. S. Oguz, and M. Toussaint, "Long-horizon multi-robot rearrangement planning for construction assembly," *IEEE Transactions on Robotics*, 2022.

[17] A. Solano, A. Sieverling, R. Gieselmann, and A. Orthey, "Fast-drrt*: Efficient multi-robot motion planning for automated industrial manufacturing," *arXiv preprint arXiv:2309.10665*, 2023.

[18] J. Chen, J. Li, Y. Huang, C. Garrett, D. Sun, C. Fan, A. Hofmann, C. Mueller, S. Koenig, and B. C. Williams, "Cooperative task and motion planning for multi-arm assembly systems," *arXiv preprint arXiv:2203.02475*, 2022.

[19] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects: Steven m. lavalle, iowa state university, a james j. kuffner, jr., university of tokyo, tokyo, japan," *Algorithmic and computational robotics*, pp. 303–307, 2001.

[20] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol. 12, no. 4, pp. 566–580, 1996.

[21] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete rrt for exploration of implicit roadmaps in multi-robot motion planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 501–513, 2016.

[22] 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.

[23] I. Solis, J. Motes, R. Sandström, and N. M. Amato, "Representation-optimal multi-robot motion planning using conflict-based search," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4608–4615, 2021.

[24] J. Kottinger, S. Almagor, and M. Lahijanian, "Conflict-based search for multi-robot motion planning with kinodynamic constraints," in *Proc. of the IEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2022, pp. 13 494–13 499.

[25] J. P. Van Den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in *Proc. of the IEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2005, pp. 430–435.

[26] K. Okumura and X. Défago, "Quick multi-robot motion planning by combining sampling and search," *arXiv preprint arXiv:2203.00315*, 2022.

[27] G. Han, J. Park, and C. Nam, "Stop-n-go: Search-based conflict resolution for motion planning of multiple robotic manipulators," *arXiv preprint arXiv:2410.07606*, 2024.

[28] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 7643–7650.

[29] E. Boyarski, A. Felner, R. Stern, G. Sharon, O. Betzalel, D. Tolpin, and E. Shimony, "Icbs: The improved conflict-based search algorithm for multi-agent pathfinding," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 6, no. 1, 2015, pp. 223–225.

[30] J. Li, W. Ruml, and S. Koenig, "Eecbs: A bounded-suboptimal search for multi-agent path finding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 14, 2021, pp. 12 353–12 362.

[31] G. Mouratidis, B. Nebel, and S. Koenig, "Fools rush in where angels fear to tread in multi-goal cbs," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 17, 2024, pp. 243–251.

[32] K. Hauser and J.-C. Latombe, "Multi-modal motion planning in non-expansive spaces," *International Journal of Robotics Research*, vol. 29, no. 7, pp. 897–915, 2010.

[33] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. of the IEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 14–18 Sept. 2014, pp. 2997–3004.

[34] A. Atias, K. Solovey, O. Salzman, and D. Halperin, "Effective metrics for multi-robot motion-planning," *International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1741–1759, 2018.

[35] M. Faroni, N. Pedrocchi, and M. Beschi, "Accelerating sampling-based optimal path planning via adaptive informed sampling," *Autonomous Robots*, vol. 48, no. 2, pp. 1–12, 2024.

[36] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[37] M.-C. Kim and J.-B. Song, "Informed rrt* with improved converging rate by adopting wrapping procedure," *Intelligent Service Robotics*, vol. 11, pp. 53–60, 2018.

[38] R. Geraerts and M. H. Overmars, "Creating high-quality paths for motion planning," *International Journal of Robotics Research*, vol. 26, no. 8, pp. 845–863, 2007.

[39] Z. Kingston, C. Chamzas, and L. E. Kavraki, "Using experience to improve constrained planning on foliations for multi-modal problems," in *Proc. of the IEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2021, pp. 6922–6927.

[40] V. N. Hartmann, M. P. Strub, M. Toussaint, and J. D. Gammell, "Effort informed roadmaps (EIRM*): Efficient asymptotically optimal multi-query planning by actively reusing validation effort," in *International Symposium of Robotics Research*, 2022.