A Benchmark for Multi-Modal Multi-Robot Multi-Goal Path Planning

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

***Abstract*— In many industrial robotics applications, multiple robots are working in a shared workspace to complete a set of tasks as fast as possible. Such settings can be treated as multi-modal multi-robot multi-goal path planning problems, where each robot has to reach *a set* of goals. Existing approaches to this type of problem are neither optimal nor complete. We tackle this problem as a single centralized path planning problem and present planners that are probabilistically complete and asymptotically optimal. The planners plan in the composite space of all robots and are modifications of standard sampling-based planners where we introduce the required changes to work in our multi-modal, multi-robot, multi-goal setting. We validate the planners on a diverse range of problems including scenarios with various robots, planning horizons, and collaborative tasks such as handovers, and compare the planners against a suboptimal prioritized planner.**

Videos and code for the planners and the benchmark is available at https://github.com/vhartman/multirobot-pathplanning-benchmark.

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 are also 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 independently 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 start moving 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, or picking and placing multiple things after another in order to sort objects. Even when only considering a single pick and place task per robot, each robot needs to reach two goals: The pick, and the place location. Since the robots act in the same environment in these scenarios and thus possibly block each other from doing their tasks, 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.

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.

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

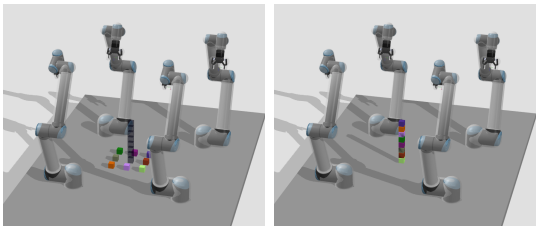


Fig. 1: Example of a typical problem of four robots cooperating to stack boxes with (left) initial state and (right) goal state.

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		✓	✓	✓	✓	✓	✓

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

Our contributions in this paper are

- a formalization of multi-modal, multi-robot, multi-goal path planning in continuous spaces,
- adaptions of probabilistically complete and almost-surely asymptotically optimal planners to this setting.

In addition, 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.

II. Multi-Robot Multi-Goal Path Planning

Informally, our objective is to find a collision free path for each robot that passes through a sequence of goals that minimizes a cost (e.g., the latest completion time of all robots). Goals might involve multiple robots: A handover of an object between two robots implies a constraint on two robots. A goal could additionally 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 et al. [4], and generalize it to multiple robots. Compared to [4], we *do not* explicitly 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 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 of a task can be fulfilled by a set of poses, where the chosen pose 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, which 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* $m \in \mathcal{M} = \mathcal{S} \times \mathcal{Q}_o$ to refer to the combination of the discrete task assignment s (which implies a scene-graph), and the poses of all movable objects $q_o \in \mathcal{Q}_o$, defined by the relative transformation to their parent-frames.

3) Task Order Specification: A task sequence is induced by an oracle $\mathcal{O}(m) : \mathcal{M} \rightarrow \mathcal{S}^*$ which maps the current assignment and state of the environment to the list of possible task assignments $\mathcal{S}^*$. A (partial) task sequence implies all possible transitions between task assignments, and thus all logically valid task assignment sequences. We use $\mathcal{T}$ to denote the set of all logically valid task assignment sequences that bring us from the start to the goal and obey the constraints imposed by the oracle.

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

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

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 $(R, \mathcal{Q}_R, q_{\text{start}}, m_{\text{start}}, \mathcal{O})$, where R is the set of robots, $\mathcal{Q}_R$ is the configuration space of the robots, q_{start} is the initial state, m_{start} is the start mode, and $\mathcal{O}$ is the oracle giving us the possible next tast assignments from the current mode. In the following, we will use q^{r_i} for the pose of robot r_i.

We want to find a collision free path $\pi(t) : \mathbb{R} \rightarrow \mathcal{Q}_R$ and the task assignment sequence $s(t) : \mathbb{R} \rightarrow \mathcal{S}_R$, that minimizes the cost function $c(\cdot)$. The path π maps time to the composite 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.

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

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

where q_1 and q_2 are two poses, and q^r refers to the pose of robot r. If w is small, we get a minimum makespan optimization problem (which is 'regularized' by the path length), and if w is 1 the total cost is the sum of all robot path lengths.

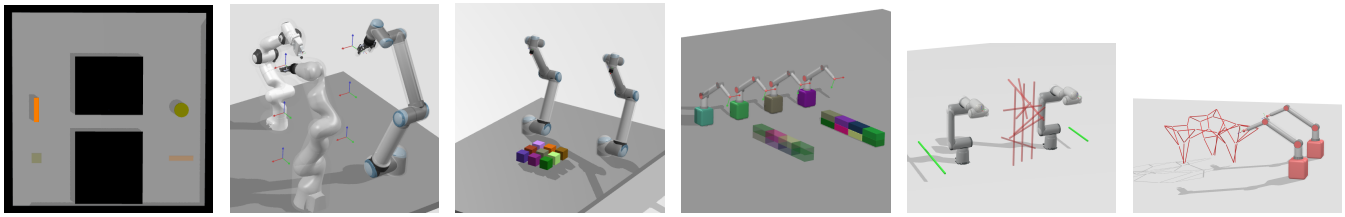
III. Planners

We adapt four sampling-based planners (RRT*, PRM*, EIT*, AIT*) to plan in the space introduced in the previous section. The main adaptions that are necessary are the distance function to connect only *within a mode*, and how we sample the space that we previously defined. Put briefly, we maintain a set of reached modes, and sample a mode from this set uniformly at random. The continuous configuration is sampled as in the standard versions of the planner.

IV. Experiments

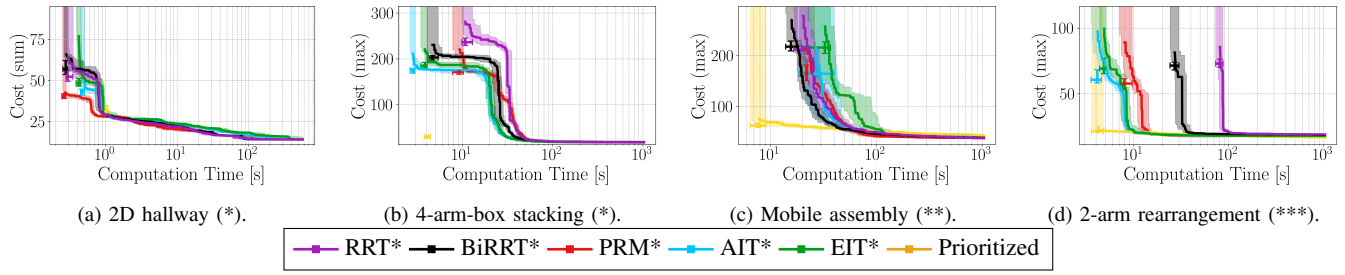
The main contribution of this work is the formulation of the multi-modal, multi-robot, multi-goal planning problem and the adoption of standard sampling-based planners to the formulation. We implement this formulation in a variety of base scenarios in an open-source benchmark featuring diverse sets of 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 such as assemblies of architectural artifacts [18], [19] or long horizon rearrangement problems with up to 5 robots (with a total of 30 degrees of freedom), and up to 74 goals. For many of the problems, there are

[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.



(a) 2D hallway. (b) Multi waypoints. (c) Rearrangement. (d) Mobile assembly. (e) Architectural artifact. (f) Truss structure.

Fig. 2: 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. All scenarios support different task-specifications.



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

Fig. 3: Evolution of median cost over time along with the 95% non-parametric confidence intervals over 50 runs. We also show the median initial solution time and cost using the square with error bars. (*) indicates a fully given task sequence, (**) indicates a partial task ordering, (***) indicates unassigned and unordered tasks. Note that the prioritized planner only acts as anytime planner and improves its cost in the (**) and (***) settings, where multiple different sequences can be generated and planned for.

versions with a complete task ordering and assignment, and others that are either unordered, partially ordered, or where tasks are unassigned. We show a selection of base-scenarios in Fig. 2.

We provide proof-of-concept implementations of the planners in Python, while the computationally expensive parts, i.e., collision checking and forward kinematics use a more performant backend, which can be easily replaced, allowing for, e.g., GPU parallelization if the backend supports it. Currently, the benchmark supports implementations of environments in Pinocchio [20] and RAI[2].

A. Experiments

We present a selection of experiments to showcase the different types of scenarios and task sequence specifications and compare the optimal planners to a prioritized planner [21] on them[3]. We report results for four representative scenarios: (a) A 2D scenario with two robots that is similar to the classic wall-gap, where the robots have to switch positions and go back to their start poses (6D conf. space, 3 subgoals, Fig. 2a), (b) a scenario with 4 robot arms where 8 boxes have to be stacked (24D conf. space, 17 subgoals, Fig. 1), (c) a scenario involving 4 mobile manipulators rearranging a wall (24D conf. space, 17 subgoals, Fig. 2d), and (d) a scenario with 2 robot arms that have to collaborate with handovers to rearrange and reorient 9 boxes (12D conf. space, 28 subgoals, Fig. 2c). In these scenarios, the task sequence is fully determined for scenario (a) and (b), *partially given* for scenario (c) and *neither assigned nor ordered* for scenario (d). To deal with partially ordered problems in the prioritized planner, we generate random sequences, and plan for those until the time runs out.

We use the cost function with $w = 0.01$, i.e., the min-makespan problem with a small path-length regularization in scenarios (b)-(d), and $w = 1$ in scenario (a). We report the median solution costs over 50 runs with different random seeds along with confidence intervals. We use a greedy mode sampling strategy until a path is found, and then switch to uniform mode sampling in all planners if not stated otherwise.

The experiments were run with Python 3.10 on a Ryzen 7 5800X (8-core, 4'491 MHz) and 32 GB RAM.

B. Results

Figure 3 shows the resulting cost evolution plots. Comparing the optimal planners introduced in this work and the prioritized planner shows that the suboptimal planner finds competitive solutions quickly, particularly in the robotic problems. As expected, the optimal planners converge to solutions with lower cost, but do so at a later time. Surprisingly, the planners that plan in composite space find initial solutions faster than the suboptimal planner in some settings. We explain this mainly with the fact that treating the previously planned paths as fixed sometimes constrains the planning space such that the planning problem becomes considerably harder compared to the setting where all agents can be moved around freely.

Convergence in complex problems: In scenarios (c) and (d) the planners do not reliably converge to the same solution at the end of their runtime, showing that the found solutions are often not optimal. This is because the problem space is

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

[3] Synchronized planners and standard MAPF planners are not applicable to these problems, and are thus not part of the benchmark.

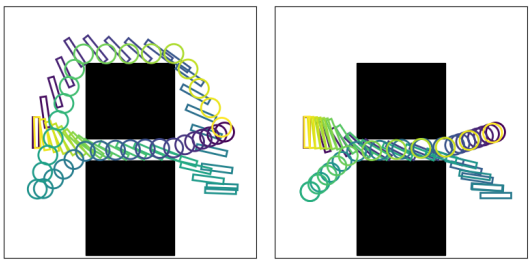


Fig. 4: The optimal paths when using a max **(left)** or sum **(right)** cost function in the 2D hallway scenario, where the robots have to reach a goal on the other side and return. Color indicates time from purple (start) to yellow (end).

extremely large due to the discrete choice of task assignment and ordering in addition to the high dimensionality of the continuous space. Thus, while the planners are theoretically optimal, in practice, we find that the planners tend to get stuck in a local optimum and do not switch the discrete assignment and ordering often. We also want to note here that in scenario (c), the median final cost of the prioritized planner is lower than the median final cost of some of the composite space planners. This is due to the better exploration of different sequences.

Influence of the cost function: We illustrate the difference of the sum-cost and the max-cost on the hallway example in Fig. 4: 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.

V. Discussion & Limitations

We presented a formalization of the multi-modal, multi-robot, multi-goal motion planning problem, and open-source the benchmark we developed, containing diverse problems reaching from simple environments with short goal sequences, to relatively long horizon problems requiring coordination of multiple robots. We adapt multiple standard sampling-based planners that are probabilistically complete and asymptotically optimal to this type of problem. We also benchmarked a prioritized planner against optimal planners and show that the solutions found by such a suboptimal planner can be competitive in some settings.

Due to planning in the composite space and treating the problem as a single big planning problem, the planners do not scale well with the number of robots (due to the increase in state-space dimensions) and they do not scale well with the number of goals, as can be seen in the slower convergence in the high dimensional problems. However, we believe that formulating the problem in composite space helps understand how choices such as prioritization affect the continuous space, and this deeper understanding can enable the design of better planners.

There are many extensions to the chosen problem formulation: The formulation and planners we propose supports extension of the transition logic to full multi-robot task and motion planning by changing the task-assignment-oracle. In the future, we also want to support constrained multi-robot planning, or kinodynamic planning.

References

[1] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artif. intell.*, 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. Robot. Automat. (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 Robot. and Automat. Lett. (R-AL)*, 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. Robot. Automat. (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 Trans. Robot.*, vol. 32, no. 5, pp. 1163–1177, 2016.

[9] R. Stern, "Multi-agent path finding–an overview," *Artificial Intelligence: 5th RAAI Summer School, 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 *Conf. on 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 *Int. Conf. on Autom. Plan. and Sched.*, 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. Robot. Automat. (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. Robot. Automat. (ICRA)*, vol. 2, 2002, pp. 2112–2119 vol.2.

[15] A. Orthey, S. Akbar, and M. Toussaint, "Multilevel motion planning: A fiber bundle formulation," *Int. J. of Robot. 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 Trans. Robot.*, 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] E. M. Skevaki, M. Kladeftira, Z. Wang, and S. Parascho, "Human-robot interaction - workshop at design modeling symposium 2024," sep 2024. [Online]. Available: https://infoscience.epfl.ch/handle/20.500.14299/253264

[20] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE Int. Symp. on Syst. Integrations (SII)*, 2019.

[21] V. N. Hartmann and M. Toussaint, "Towards computing low-makespan solutions for multi-arm multi-task planning problems," in *Int. Conf. on Autom. Plan. and Sched.: Planning and Robotics Workshop (RobPlan)*, 2023. [Online]. Available: https://arxiv.org/abs/2305.17527