2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

R-LGP: A Reachability-guided Logic-geometric Programming Framework for Optimal Task and Motion Planning on Mobile Manipulators

Kim Tien Ly[1], Valeriy Semenov[1], Mattia Risiglione[2], Wolfgang Merkt[1], Ioannis Havoutis[1]

***Abstract*— This paper presents an optimization-based solution to task and motion planning (TAMP) on mobile manipulators. Logic-geometric programming (LGP) has shown promising capabilities for optimally dealing with hybrid TAMP problems that involve abstract and geometric constraints. However, LGP does not scale well to high-dimensional systems (e.g. mobile manipulators) and can suffer from obstacle avoidance issues due to local minima. In this work, we extend LGP with a sampling-based reachability graph to enable solving optimal TAMP on high-DoF mobile manipulators. The proposed reachability graph can incorporate environmental information (obstacles) to provide the planner with sufficient geometric constraints. This reachability-aware heuristic efficiently prunes infeasible sequences of actions in the continuous domain, hence, it reduces replanning by securing feasibility at the final full path trajectory optimization. Our framework proves to be time-efficient in computing optimal and collision-free solutions, while outperforming the current state of the art on metrics of success rate, planning time, path length and number of steps. We validate our framework on the physical Toyota HSR robot and report comparisons on a series of mobile manipulation tasks of increasing difficulty. Videos of the experiments are available here.**

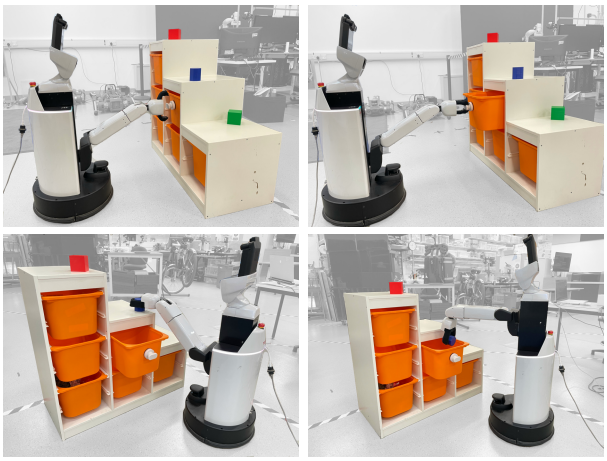


Fig. 1: Physical HSR performing R-LGP solution for table clearing task. The captured actions include grabbing knob, opening/closing drawer, picking object, and dropping object.

I. Introduction

Task and motion planning (TAMP) is the process of decision making that takes a given task as input and outputs a sequence of robot configurations to complete the task. While task planning focuses on high-level task-oriented strategies, motion planning refers to low-level control algorithms that determine the feasibility of robot motion and the continuity considering actuation and joint limits, environmental obstacles, or uncertainties. These methods take into account the variations of sequences of actions that can lead to the desired goal. Therefore, TAMP planners must address both task-level and motion-level requirements, which can be solved either independently or jointly.

A major challenge in TAMP is that high-dimensional geometric constraints in motion control (kinematics, joint limits, reachability, etc.) can limit the effectiveness of high-level strategies. The problem becomes more complex with long-horizon tasks, higher-degrees-of-freedom robots, or multiple–possibly dynamic–obstacles. Therefore, the deliberative function (high-level task planner) should either be informed or have sufficient restrictions on the robot's physical limitations to produce feasible solutions. Consequently, TAMP requires the integration of both high- and low-level planning and TAMP research generally aims to effectively combine both artificial intelligence techniques in task planning and advanced motion planning to tackle such problems.

In this paper we propose a novel TAMP approach to solve sequential decision-making applications inheriting logic-geometric programming (LGP) [1]. Our framework generates optimal plans for complex robot tasks, in contrast to current approaches that generate solely feasible solutions, for example, the well-known TAMP solver PDDLStream [2]. Our approach tightly integrates a reachability graph with LGP and allows us to achieve robust, collision-free and kinematically-efficient solutions to LGP on high-dimensional mobile manipulators. We explore the efficacy of our framework in a range of existing TAMP problems of increasing difficulty and report results on success rates, planning times and path lengths. Furthermore, we validated on a physical robot, the Toyota Human Support Robot (HSR), aiming at realistic and practical applications, where geometry and kinematics must be respected.

Our main contributions are (1) a LazyPRM-inspired graph for motion planning on floating-base manipulators, (2) a reachability graph serving as symbolic and motion guidance for the LGP planner, (3) a kinematically-effective optimization-based system for TAMP on mobile manipulators. Our novel TAMP approach is a Reachability-guided Logic-geometric Programming framework, in short, *R-LGP*.

[1]K. T. Ly, V. Semenov, W. Merkt and I. Havoutis are with the Dynamic Robot Systems (DRS) group, Oxford Robotics Institute, University of Oxford. Email: {ktien,valeriy,wolfgang,ioannis}@robots.ox.ac.uk.

[2]M. Risiglione is with Istituto Italiano di Tecnologia. Email: mattia.risiglione@iit.it.

II. Related work

A. Satisfactory TAMP

The main challenge in combining task and motion planning is their hybrid nature. While task planning involves discrete task specifications, motion planning is typically solved in continuous space. Task and motion planning are traditionally combined on a level basis, which can be decoupled or integrated. Accordingly, each layer will keep its own level of abstraction and the TAMP model should be able to translate knowledge between the two levels. Given the introduction of STRIPS (Stanford Research Institute Problem Solver) [3], robot planning was initially studied with a decoupled hierarchical structure [4]. The approach in this early work assumes that motion control can solve all high-level actions without alternative backtrack solutions. Meanwhile, integrated TAMP [5] considers failures and can replan to ensure that the strategy is executable. Common methods are taking one of the two parts as the base plan and solving the other level accordingly. For example, Srivastava et al. [6] introduces a TAMP method that backtracks and finds alternative task plans if the motion solver fails. Such approach usually discretizes the continuous space for symbolic planning. The method can utilize off-the-shelf planners and state-of-the-art techniques. TMKit [7] is introduced as the first open-source framework to implement such a structure, based on [8][9]. There is an opposite approach where the task search space is bounded by sampling the continuous workspace [10]. Recently, Kim et al. [11] introduced a reachability tree-based sampling algorithm that pre-generates goal state to bias action search. Given that task planning is usually in a finite domain, having to re-plan the task sequences is considered to produce a lower cost compared to continuously checking geometrical feasibility.

PDDLStream [2] extends Planning Domain Definition Language (PDDL)[12] by using streams to enable sampling procedures. This work proves to be a modular and domain-independent approach to TAMP.

B. Optimal TAMP

To reduce complexity, TAMP approaches usually tackle task and motion planning separately and focus more on completeness. Solving such problems optimally is difficult because the optima of each layer might not be the overall optimal solution. An example can be found in [13], where the authors used a two-layer hierarchical structure of optimization to find asymptotically optimal solutions to coverage planning task. Kongming [14] was the first approach to integrate continuous control variables in an activity planner. The method combines a Planning Graph for discrete actions and Flow Tubes for continuous actions, which was modeled as a mixed logic linear (non-linear) program. The method ties to a fixed time discretization, which is difficult to apply on long-horizon tasks. However, it brings up the need for a tightly coupled task and motion planning framework in robotics applications.

Toussaint [1] introduced the logic-geometric programming (LGP) algorithm, which formulates TAMP as a mathematical program that uses optimization methods to find locally optimal plans instead of solely feasible ones. The task in this work is to build the highest stable tower from a list of blocks and boards. By bringing logic into geometry, LGP does not need to arbitrarily discretize the continuous space and directly optimize continuous solutions. An extended work from the team [15] integrated RRT to reconstruct a large pavilion. Although the method solves the issue when the optimizer fails in a cluttered environment, the system ignores system uncertainties. Variants of LGP papers include a heuristics method for long-horizon tasks [16][17], a dynamic LGP for human motion prediction [18] or an approximation solver for cooperative manipulation [19]. Other optimization-based mathematical approaches to TAMP is constraint programming (CP) [20] [21] and mixed integer programming (MIP) [22] [21] [23]. CP is a more general term than MIP, Booth et al. [21] did a comparison between MIP and CP and concluded that inference-based CP is better than relaxation-based MIP in terms of time and solution quality. This work is tested on simulation and claims to be case-sensitive. MIP is originally a decision making and scheduling method that solves non-convex problems. When it comes to robotics planning, this optimization solution can help to capture discrete decisions with integer variables. Deits et al. [24] introduced a planner that uses MIP to generate globally optimal sequences of footsteps in difficult terrain. Multi-robot task planning is a common application of MIP [25][26]. In short, these mathematical techniques allow encoding logical decisions and geometric constraints in nonlinear optimization models without backtracking, targeting globally optimal strategies.

As more interest has been driven to solving TAMP optimally, besides mathematical programming, a recent work from Thomason et al. [27] proposes asymptotically optimal decision-making using informed tree search. The model effectively combines constraint-based symbolic planning, distance-based predicate representation, and batch-sampling-based optimal motion planning to solve a hybrid state space problem. Recently, inspired by LGP, Sleiman et al. [28] introduced an offline bilevel optimization planner that solves multi-contact problems on loco-manipulation system. The system successfully plans whole-body motion for a quadrupedal mobile manipulator to open/close doors and dishwashers.

III. Proposed framework overview

In LGP, the authors solve TAMP with three levels of approximation and switches between symbolic search and configuration optimization that is conditioned by symbolic decisions. While *level 1* decides the symbolic sequence, *levels 2* and *3* optimize keyframes (e.g. grasp poses) and full path respectively. An extended work, RHH-LGP [16], proposes a heuristic to guide the symbolic search in level 1. The kinematic reachability here, which is an important factor to the feasibility of the action sequences, is fixed or calculated using Euclidean distance or the length of robot's links. In this work, we extend LGP with a sampling-based reachability graph to enable solving optimal TAMP on high degrees-of-freedom (DoF) mobile manipulators. The proposed reachability graph can also incorporate environmental information (obstacles) to provide the planner with sufficient geometric constraints. This reachability-aware heuristic efficiently prunes infeasible sequences of actions

Algorithm 1 Reachability-guided LGP

Require: symbolic goal g, initial symbolic state s_0, initial configuration x_0, world W

$s \leftarrow s_0$
while not $s \in S_{goal}(g)$ **do**
  PathFound $\leftarrow$ False
  SymbolicSearch $\leftarrow s$
  Switch $\leftarrow \emptyset$
  Path $\leftarrow \emptyset$
  while not PathFound **do**
    // Construct reachability graph
    RG = constructRG(W)
    // Symbolic search
    $n \leftarrow$ SymbolicSearch.argmin(heuristicCost(**RG**))
    Switch.append(n)
    if not $n \in S_{goal}(g)$ **then**
      SymbolicSearch.append(n.expand())
    else
      // Optimize over kinematic switches
      if SwitchOptimization(Switch) **then**
        waypoints $\leftarrow$ getWaypoints(**RG**, n)
        Path.append(waypoints)
        // Optimize over the full path
        if PathOptimization(Path) **then**
          PathFound $\leftarrow$ True
          $s \leftarrow n$
      else
        break

in the continuous domain. Hence, it reduces replanning by securing feasibility at the final full trajectory optimization.

Our reachability-guided LGP (R-LGP) pipeline is described in Algorithm 1. The proposed reachability graph serves as a guide to the first (symbolic) and final (full path) layers in LGP. In the pipeline, nodes chosen from the symbolic search, with the help of our reachability heuristic, will be sent to the kinematic switch optimization. Once a feasible sequence is found, the system will solve the full path optimization with the guided path from the precomputed reachability graph. The main contribution of the graph is twofold: *a)* to provide a heuristic that informs the LGP's symbolic planner about kinematics and geometry; *b)* to provide collision-free guidance to the LGP's trajectory optimization at level 3 (full path), in order to avoid local optima.

IV. Reachability Graph

The reachability graph is designed in a LazyPRM-like [29] manner. We compute the graph with node validation and edge checking during planning.

A. Graph generation

This section explains the **constructRG** function in Algorithm 1.

1) Node sampling: Nodes are randomized in task space $x \in \mathbb{R}^3$ with uniform distribution $U(p_{\min}, p_{\max})$. Nodes in the graph represent end-effector positions of the mobile manipulator, which sufficiently denotes robot reachability. We validate nodes with a constrained optimization formulation. The robot model is defined as a loco-manipulation system with a floating-base torso as in Eq. 1, where m is the number of DoFs of the manipulator. This allows the framework to be implemented on either legged or wheeled mobile platforms. In our work, the mobile platform is holonomic and does not have any restrictions on translation or rotation, however, these can be added in as constraints.

$$\begin{aligned} q &= [q_{\text{base}}, q_{\text{manipulator}}], \\ \text{where } q_{\text{base}} &= [x_{\text{base}}, y_{\text{base}}, \psi_{\text{base}}], \\ q_{\text{manipulator}} &= [q_1, q_2, q_3, ..., q_m]. \end{aligned} \tag{1}$$

The nonlinear optimization formulation for node validation is defined in Eq. 2, where x_0 is the checked node, representing the reference for the robot end-effector position. $g_{\text{prec}}()$ checks collision between the inspecting node and the environment, and determines the precondition for the optimization with a constant $M \rightarrow +\infty$. This condition removes unreachable nodes for the sake of processing time. On the other hand, $f_{\text{kin}}()$ computes the end-effector position x that corresponds to the joint values q. Function $g()$ defines inequality constraints for whole-body joint limits and collision. In the reachability graph, orientation of the end-effector is not constrained for flexibility and generality. Our graph is designed to generate a guiding cost and path for LGP. The full trajectory optimization, with orientation constraints, will then be computed on the final layer with constraints for the relevant manipulation action.

$$\begin{aligned} \min_{q} \; & M \cdot \max(0, g_{\text{prec}}(x_0)) + |x - x_0|^2 \\ \text{s.t. } & g_{\text{prec}}(x_0) \leq 0, \\ & x = f_{\text{kin}}(q), \\ & g(q) \leq 0. \end{aligned} \tag{2}$$

After collision and inverse kinematics (IK) validation is performed, the node is added to the graph's node list N as $n(x, q, c)$. While $x \in \mathbb{R}^3$ is the end-effector of the robot, $q \in \mathbb{R}^{3+m}$ denotes the optimized IK configuration. c equals to the cost value from the optimization solution. The sampling process stops when the number of N reaches a predefined number N_{node}.

2) Edge connections: As mentioned, we relax edge connection by assuming all edges are collision-free. For each sampled node in the list N, we connect to k nearest neighbours without checking for collision. Each edge is added to the graph with the cost defined in Eq. 3. The cost includes a combination of weighted Cartesian and configuration Euclidean distances between two end nodes, along with nodes' costs. w_x, w_q and w_c are the correspondent weights for the Cartesian, configuration distance and IK cost.

$$c_e = w_x \cdot \|x_1 - x_2\| + w_q \cdot \|q_1 - q_2\| + w_c \cdot (c_1 + c_2) \tag{3}$$

B. Path querying

1) Solution library: The reachability graph runs as an underlying service, waiting for the LGP planner to query two ends of a path. The service leverages a library to store previously computed solutions to avoid replanning for the same path and save planning time. During symbolic search in the first LGP's layer, the planner might iterate through different logical sequences and query a path in opposite

directions. For instance, a first call may seek a path from A to B, and subsequently, a solution from B to A. This is beneficial for tasks at the same levels, where one does not have a precondition on another action's completion.
The structure of each solution is shown in (4), where *key* contains the start and end point of the path, *path* and *cost* come from the resulting solution with n being the number of nodes on the path. Noted that the final configuration in *path* q_{1n} matches x_2 Cartesian pose. Each *key* is unique across the library in sorted order, and is used to query the data bidirectionally. On each call, the service checks the library and returns the stored solution. If the requested *key* is new to the library, the system starts to query the graph for the path. The answer is then added to the library for future reference.

$$\begin{aligned} \text{solution} = \{&\text{key: } (x_1, x_2)\ , x_i \in \mathbb{R}^3 \\ &\text{path: } [q_{10}, q_{11}, ..., q_{1n}] \in \mathbb{R}^{n\times(3+m)} \\ &\text{cost: } h\} \end{aligned} \tag{4}$$

In this data structure, *cost* is returned if the pipeline is querying heuristic cost, which is the function **heuristicCost** in Algorithm 1. When the framework reaches the final layer, *path* serves as guidance for trajectory optimization, accessed through the function **getWaypoints**.

2) Path planning: The reachability graph planner receives two Cartesian positions for each path querying message *key* as starting point and ending point, naming x_k and x_{k+1}. Nodes corresponding to x_k and x_{k+1} are added to the graph upon solving (2). Collision is checked for connecting these two nodes to the existing graph. If either x_k or x_{k+1} is unable to connect to the graph, we start enhancing the node. Gaussian sampling is applied with in-loop decreased covariance to find adjacent nodes as waypoints to connect to the existing graph. During this process, collision checking on both nodes and edges is performed to ensure a meaningful enhancement.

We use Dijkstra for finding the shortest path in the graph. We verify the path by checking collisions on all connected edges. We linearly interpolate the trajectory in configuration space with a predefined number of steps L. An edge with a detected collision is removed from the graph and replanning is triggered. In the case that any node loses all connections due to edge removal, it will be enhanced with the same method as starting/ending node enhancement. The difference is that the inspecting node can be replaced with an effective sampled one, which has the lowest IK cost c and is able to connect to the graph. In this sense, we ensure that the graph is fully connected, meaning that no subset is disconnected from the rest of the graph. During planning, the system updates the graph with enhancement and edge removal, benefiting future planning queries.

The flowchart in Fig. 2 shows in detail the path planning algorithm along with the interactions between the reachability graph and the LGP framework. The heuristic cost h is used internally in the symbolic search level. Before entering into the last stage of LGP, we interpolate $[q_k, q_{k+1}]$ ($k \in [0, T-1]$) with corresponding $[q_{k0}, .., q_{kn}]$ as input guidance for full path optimization.

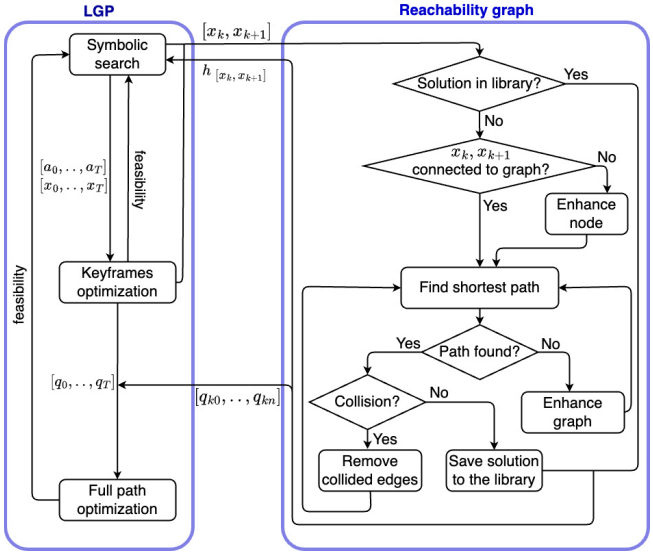


Fig. 2: Path querying system in R-LGP.

V. Evaluation

A. Simulation evaluation

In order to validate our approach, we propose a set of simulations and hardware experiments, performed on the PR2 and HSR robots. Our baseline for comparison is the heuristic-based LGP - RHH-LGP [16]. In addition, we compare our system against PDDLStream [2], the most popular and ready-to-use TAMP solver. We implemented the adaptive version of PDDLStream, which claims to outperform all other approaches for cost-sensitive problems. This allows us to set a priority on minimizing path length. There is an *optimal* flag in PDDLStream solver which determines whether or not the planner should explore more options and conclude with the best solution. We name *PDDLStream nonoptimal* and *PDDLStream optimal* for PDDLStream with *optimal* set to false and true respectively.

The experiments are designed to address the ability in accommodating geometric and kinematic constraints in integrated TAMP problems. We conducted an extensive evaluation with 100 runs in simulation for each experiment. All random tasks are intentionally feasible. The recorded metrics are:

- *Success rate*: the number of collision-free TAMP solutions over the total number of runs.
- *Planning time*: average time taken to get to successful solutions.
- *Path length*: average length, in meters, of the robot's base trajectory, assuming a floating-base robot.
- *Number of steps*: average number of steps to complete the given task. For this metric, we only consider key switches, e.g. pick, place, which intuitively correspond to the number of objects.

1) Pick and place: In this scenario, the PR2 robot is asked to pick and place 3 objects on the tables to the tray. This is a default TAMP task in the PDDLStream framework and is widely used in TAMP development [2]. The objects are at random positions on the table over 100 runs. We set a fixed

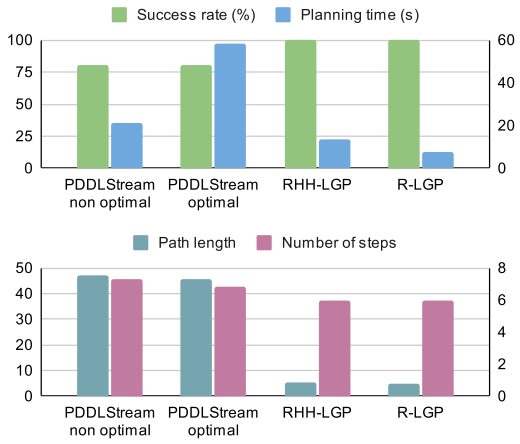


Fig. 3: Comparison result of the TAMP planners on pick and place task.

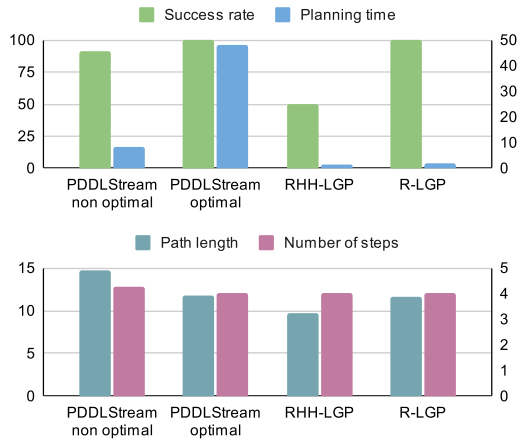


Fig. 4: Comparison result of the TAMP planners on sorting task.

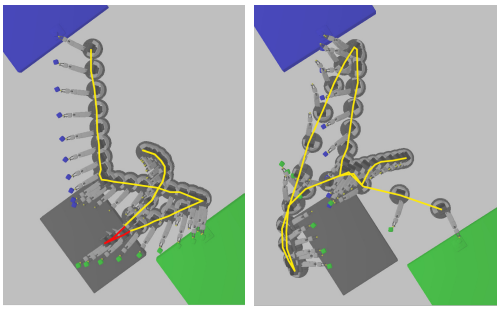
table size and all random poses are reachable from one side of the table.

The results in Fig. 3 show that the PDDLStream optimal solution is better in path length and step count than the default PDDLStream, although the latter achieves a shorter planning time. With PDDLStream solutions, PR2 tends to travel to the side of the table nearest to the object to pick it up regardless of reachability. LGP frameworks provide shorter trajectories by not navigating around the table, hence, also reduce the execution time. Apart from optimality, as opposed to PDDLSream, LGP frameworks achieve 100% success rate. Our R-LGP framework effectively reduces planning time leveraging the reachability heuristic.

2) Sorting: We increase the complexity of both task planning and motion planning in this task. The robot is asked to sort objects into trays with the same colors. Unlike the classic sorting task, we extend the experiment to address the reachability and mobility awareness of the system. In particular, we use larger table tops in this test, and randomized objects can be out-of-reach from one side of the table. This task requires TAMP solvers to accommodate obstacle-free configuration trajectories in solving high-level tasks, specifically requiring the robot to navigate around the table. To ensure all tasks are feasible, we constraint the objects' randomized locations to be within reachable distance from at least one side of the table.

In Fig. 4, it is worth noticing that the baseline RHH-LGP fails to plan a collision-free trajectory in 50% of the cases. Due to local optima in trajectory optimization, RHH-LGP planner can only solve tasks where the randomized location is within reach from the same side to the current position. This issue then filters runs with shorter path and results in a shortest average path length. PDDLStream optimal and R-LGP provide similar quality of final solutions with the same success rate of 100%. However, the planning time in R-LGP is significantly reduced from 48s to 2s.

Fig. 5 captures an example of TAMP results for the sorting task with the LGP baseline (a) and our planner (b). In the figures, the robot's base trajectory is shown in yellow line, which also denotes the resulted task sequence. In RHH-LGP,



(a) RHH-LGP (b) R-LGP

Fig. 5: TAMP results. Objects are initially spawned on the grey table and should be put on the table with the same color. R-LGP outperforms RHH-LGP in generating reachability-aware robot base trajectory (yellow lines) and avoiding collisions (red lines).

the planner considers the distance-based heuristic, hence, decides to sort the green block first. Meanwhile, our heuristic cost informs the high-level planner that the robot must go around the table in order to pick up the green block. Therefore, the decision to pick the blue block first produces a shorter travelling cost. With the help of the reachability graph, the symbolic planner can consider the cost of navigating taking into account reachability capabilities. In terms of full trajectory optimization, the red line shown in Fig. 5a is the violation of collision checking in RHH-LGP. We successfully solve this issue with the guidance from the reachability graph to the final layer in LGP, which enables the collision-free path in Fig. 5b.

B. Real robot validation

We validated our framework on a physical HSR. We implemented 2 tasks of increasing difficulty:

- *Sorting task*: the robot is asked to pick objects and place in the correct trays (Fig. 6).
- *Table clearing*: the task is to put objects from the surface into the drawer (Fig. 1).

In this real-world experiment, R-LGP proved to successfully solve a longer-horizon task with table

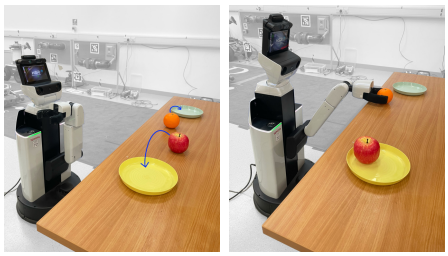


Fig. 6: Physical HSR using R-LGP solver in sorting task.

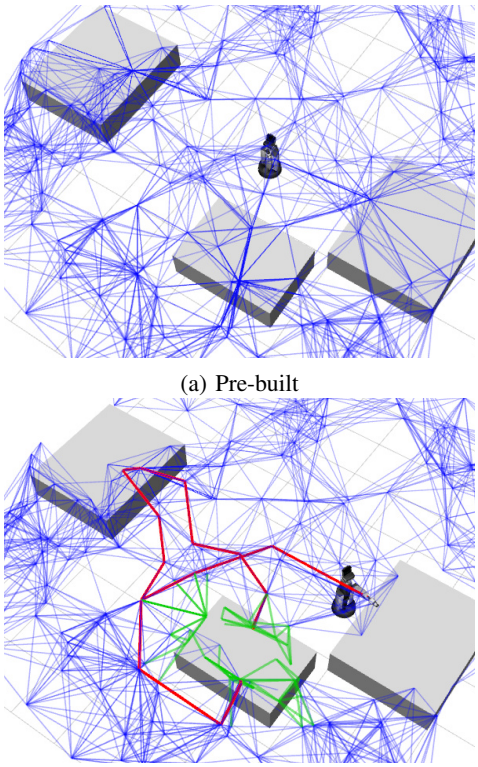
clearing. The desired symbolic sequence of this mission is (*take_knob, open_drawer, pick_object, drop_object, take_knob, close_drawer*). Figure 1 provides a visualization of the listed behaviours. Treating *take_knob* separate to *open_drawer* and *close_drawer*, the work can also be applied to opening/closing drawers, cabinets or doors. In our trial, the stair-like cabinet, as shown in Fig. 1, also poses a challenge of reachability awareness and obstacle avoidance. When the object is placed on the lower step, picking up action must consider that certain directions are blocked by the step above it.

In the sorting task, the R-LGP planner generated the result in 2s for 4 steps, with 2 objects correctly placed. It took longer for table clearing, 17s, to plan the long sequence with 6 separate steps, which the robot travelled 9m. In both examples, with our R-LGP solutions, the HSR robot can perform the task robustly with an optimal and collision-free trajectory.

C. Reachability graph evaluation

For each environment, we precompute the reachability graph only once. On average, the reachability graph is built within 14.2s with 200 samples over the specified workspaces.

We designed the sorting task to evaluate the robustness and completeness of our reachability graph, as this environment has more geometric constraints to consider during motion planning. Figure 7 visualizes our reachability graph for this scenario. The pre-built reachability graph is shown in Fig. 7a. In this step, nodes are validated with a collision-free configuration state and neighbour edges are all connected. The generated solution for the whole successful trajectory is plotted in the red line in Fig. 7b. The solution sufficiently produces a collision-free guidance path that has an understanding of the robot kinematics and reachability. This resulting path provides LGP with meaningful heuristic cost and trajectory guidance. The enhanced graph after a few queries is also denoted in Fig. 7b. The green lines represent added nodes and edges while searching for paths. In this example, the enhancement is mainly around the middle table, as objects for sorting are randomized on this table. The graph is updated over time, where edges in collision are removed and new nodes are added. This reduces planning time for subsequent queries in the same workspace.



(a) Pre-built

(b) After planning

Fig. 7: Reachability graph before and after planning. The red line is an example of the generated guided path, and green lines present the graph enhancement over time.

VI. Conclusion

R-LGP, our proposed framework, provides a robust solution to TAMP on mobile manipulators, which is tightly integrated to LGP. The introduction of the reachability graph not only successfully informs the LGP symbolic planner with information regarding reachability and mobility capabilities but also resolves local optima in trajectory optimization. The reachability-aware heuristic cost helps to prune out infeasible trajectories at the first layer, hence, reducing failures at the final full trajectory optimization. Despite being a sampling-based graph, it boosts the efficiency of both symbolic search and trajectory optimization without loss of optimality for LGP. As a result, the system is time-efficient in generating optimal TAMP solution while respecting robot geometry and kinematics. We presented an extensive evaluation to validate the framework, including real-world experiments on the Toyota HSR robot. In comparison to the state-of-the-art PDDLStream in the basic pick and place task, we reduced path length by 10 times with 6 times quicker solving time. The sorting task experiment highlighted the drawbacks of the current LGP framework, where environmental constraints can adversely affect symbolic planning. Without guidance, the baseline also fails to generate a collision-free trajectory. Hence, we achieved twice better success rate and improved completeness of the LGP solver. While sampling-based motion planning does not provide optimal solutions in limited time, our proposed reachability graph integrated to LGP can solve optimal TAMP. An avenue for future work is pruning task-specific constraints for TAMP on long-horizon applications. In our current framework, we are addressing the motion rather than task knowledge, and we believe pruning will help with scaling up to larger state spaces.

REFERENCES

[1] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[2] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 440–448.

[3] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[4] N. J. Nilsson, "Shakey the robot," *Technical note 323*, 1984.

[5] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 265–293, 2021.

[6] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 639–646.

[7] N. Dantam, S. Chaudhuri, and L. Kavraki, "The task motion kit," *IEEE Robotics & Automation Magazine*, vol. PP, pp. 1–1, 05 2018.

[8] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "Incremental task and motion planning: A constraint-based approach." in *Robotics: Science and systems*, vol. 12, 2016, p. 00052.

[9] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. Kavraki, "An incremental constraint-based framework for task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1134–1151, 2018.

[10] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 729–746, 2004.

[11] K. Kim, D. Park, and M. J. Kim, "A reachability tree-based algorithm for robot task and motion planning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 3750–3756.

[12] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "Pddl-the planning domain definition language," 1998.

[13] K. T. Ly, M. Munks, W. Merkt, and I. Havoutis, "Asymptotically optimized multi-surface coverage path planning for loco-manipulation in inspection and monitoring," in *IEEE 19th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2023.

[14] H. X. Li and B. C. Williams, "Generative planning for hybrid systems based on flow tubes," in *ICAPS*, 2008, pp. 206–213.

[15] V. N. Hartmann, O. S. Oguz, D. Driess, M. Toussaint, and A. Menges, "Robust task and motion planning for long-horizon architectural construction planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6886–6893.

[16] C. V. Braun, J. Ortiz-Haro, M. Toussaint, and O. S. Oguz, "Rhh-lgp: Receding horizon and heuristics-based logic-geometric programming for task and motion planning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 761–13 768.

[17] D. Driess, O. Oguz, and M. Toussaint, "Hierarchical task and motion planning using logic-geometric programming (hlgp)," in *RSS Workshop on Robust Task and Motion Planning*, 2019.

[18] A. T. Le, P. Kratzer, S. Hagenmayer, M. Toussaint, and J. Mainprice, "Hierarchical human-motion prediction and logic-geometric programming for minimal interference human-robot tasks," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021, pp. 7–14.

[19] M. Toussaint and M. Lopes, "Multi-bound tree search for logic-geometric programming in cooperative manipulation domains," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4044–4051.

[20] J. K. Behrens, R. Lange, and M. Mansouri, "A constraint programming approach to simultaneous task allocation and motion scheduling for industrial dual-arm manipulation tasks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8705–8711.

[21] K. E. Booth, T. T. Tran, G. Nejat, and J. C. Beck, "Mixed-integer and constraint programming techniques for mobile robot task planning," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 500–507, 2016.

[22] M. Conforti, G. Cornuéjols, G. Zambelli *et al.*, *Integer programming*. Springer, 2014, vol. 271.

[23] D. Ioan, I. Prodan, S. Olaru, F. Stoican, and S.-I. Niculescu, "Mixed-integer programming in motion planning," *Annual Reviews in Control*, vol. 51, pp. 65–87, 2021.

[24] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS international conference on humanoid robots*. IEEE, 2014, pp. 279–286.

[25] P. Culbertson, S. Bandyopadhyay, and M. Schwager, "Multi-robot assembly sequencing via discrete optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6502–6509.

[26] M. Lippi and A. Marino, "A mixed-integer linear programming formulation for human multi-robot task allocation," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021, pp. 1017–1023.

[27] 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.

[28] J.-P. Sleiman, F. Farshidian, and M. Hutter, "Versatile multicontact planning and control for legged loco-manipulation," *Science Robotics*, vol. 8, no. 81, 2023.

[29] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 521–528.