2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Optimal Grasps and Placements for Task and Motion Planning in Clutter

Carlos Quintero-Peña, Zachary Kingston, Tianyang Pan, Rahul Shome,
Anastasios Kyrillidis, and Lydia E. Kavraki

2023 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-2365-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRA48891.2023.10161455

***Abstract*— Many methods that solve robot planning problems, such as task and motion planners, employ discrete symbolic search to find sequences of valid symbolic actions that are *grounded* with motion planning. Much of the efficacy of these planners lies in this grounding—bad placement and grasp choices can lead to inefficient planning when a problem has many geometric constraints. Moreover, grounding methods such as naïve sampling often fail to find appropriate values for these choices in the presence of clutter. Towards efficient task and motion planning, we present a novel optimization-based approach for grounding to solve cluttered problems that have many constraints that arise from geometry. Our approach finds an optimal grounding and can provide feedback to discrete search for more effective planning. We demonstrate our method against baseline methods in complex simulated environments.**

I. Introduction

Manipulation is essential for robotics. To efficiently plan for manipulation, a robot must be able to reason over discrete choices (what to do) and continuous actions (how to do it). Task and Motion Planning (TAMP) [1–7] addresses these problems with layered planning; TAMP methods use task planning [8–10] to find action sequences (*plan skeletons*) that are either resolved into continuous motion by motion planning [11–13] or are declared infeasible (e.g., due to motion planning timeout), thus requiring a new plan skeleton.

To find a continuous motion for a desired action, an action's *parameters* (e.g., how an object is grasped, where an object is placed) must be determined. Finding parameters for a plan skeleton (so called as it is missing these parameters) is called *grounding*. Commonly, grounding is achieved through discretization [1] or sampling [2, 3]. However, in cluttered environments, discretization may not have sufficient resolution to find a plan, and sampling is unlikely to produce parameters that satisfy geometric constraints [14–18].

Consider Fig. 1, where the robot must set up the 8 blue chess pieces. Given the clutter, height differences of the pieces, and the robot's hand geometry, many pieces can only be grasped with certain hand poses and in a specific order. Additionally, when problems are non-monotone (i.e., require grasping a piece more than once), intermediate placement locations must be found, which is highly non-trivial in clutter [17, 18]. For example, in Fig. 1c–d, the blue rook and other shorter pieces cannot be grasped while the queen is adjacent to them—the queen must first be moved to an intermediate location. Moreover, the queen can only be placed at the goal after the shorter bishop is finalized.

All authors are affiliated with the Department of Computer Science, Rice University, Houston TX, USA {carlosq, zak, tp36, rahul.shome, anastasios, kavraki}@rice.edu. This work was supported in part by NSF RI 2008720 and Rice University Funds.

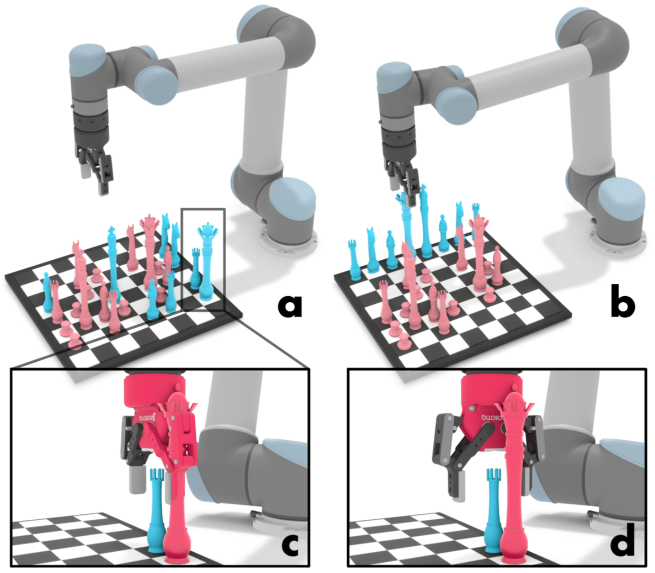


Fig. 1. A cluttered manipulation scenario. **a)** Example start configuration of the *Chessboard* problem. The robot must move all 8 blue pieces to achieve the **b)** goal. **c–d)** Example of action infeasibility from geometry. Grasping the short blue rook fails due to collision of the robot's hand and the queen (dark pink). The queen must be removed before the rook can be grasped. Our method finds these constraints and solves cluttered scenarios like this.

In this work, we focus on improving the scalability of TAMP—our contribution is a specialized grounding layer for highly cluttered environments, which we demonstrate in a TAMP framework. Our grounding layer is a mathematical program capable of finding optimal (with respect to a user-defined function) groundings for a plan skeleton. Moreover, if a feasible grounding does not exist, constraints derived from the infeasible model are added to a Satisfiability Modulo Theories (SMT) solver [8] to find a new plan skeleton. The instances of our grounding layer presented here are applicable to a limited but important class of problems typically arising in planar manipulation planning. The modeling choices in the proposed formulation (see Sec. V-C) provide global optimality guarantees in contrast to more general formulations, which we believe is critical for clutter.

In contrast to prior work, our approach does not rely on discretization of the environment [1] or approximations of free configuration space [19]. Unlike other optimization-based TAMP methods [4, 20], the mathematical models used by our method are designed for cluttered environments, and effectively guide task planning to plan skeletons that meet all geometric constraints. We demonstrate our method on several simulated manipulation tasks, all of which are cluttered and present complex geometric relationships between the objects and robot. We compare against state-of-the-art methods, showing that our method can solve cluttered problems with more objects and longer plans in less time.

II. Related Work

Our work focuses on improving scalability of TAMP methods. In TAMP, a discrete symbolic planner and continuous motion planner interact to find a plan that achieves a high-level goal [1–7]. Our method is based on using an SMT solver for TAMP, similar to [1, 21]—in these approaches, if grounding an action with motion planning fails (e.g., due to planner timeout), constraints are added to the SMT solver to generate a new plan skeleton. However, these approaches require discretized action parameters, which are represented as unique symbols in the task planner. In contrast, our method leverages constraint-based SMT solving without discretization—our approach uses abstract symbols in the task planner which are grounded through our novel optimization-based method.

Rather than discretizing action parameters, other approaches use specialized conditional samplers to ground actions in plan skeletons [2, 3, 22]. However, for highly cluttered environments, the chance of sampling feasible values is low and thus impractical. Additionally, these methods can be highly inefficient with *interdependencies* between variables in the plan skeleton—a decision made early in the plan can make it virtually impossible find feasible groundings for actions further along [23]. Our optimization-based grounding method jointly optimizes and grounds all geometric variables in a plan skeleton, avoiding these problems.

To holistically tackle the TAMP problem, the methods described in [4, 20] use nonlinear mathematical programs over a plan skeleton, which ground action parameters in stages. Similarly, our method grounds geometric variables by optimizing over a plan skeleton. However, our framework is specially suited for clutter. Under certain assumptions, our formulations can efficiently discard infeasible plan skeletons and help guiding the search towards feasible solutions by providing feedback to the task planner. Recently, [7] extended the method in [4] to discover conflicting geometric constraints within factor graphs to inform the discrete layer of a TAMP planner. In our approach, information about infeasibility from our optimization-based grounder is transformed into constraints which are added to the SMT-based task planner.

Other approaches proposed specialized motion planners with hierarchical searches to find placement poses in cluttered environments [16] or optimize grasp poses for computing fast motions [24]. Our framework computes both optimal grasp poses and placement locations in cluttered environments but it does so within a TAMP framework, which makes it useful for longer-horizon and more general robotic tasks.

A. Object Rearrangement

Object Rearrangement (OR) [15, 17, 18, 25] defines a class of TAMP problems where the goal is a desired configuration of objects. Most attention has been given to tabletop rearrangement, where objects of similar geometry on a flat

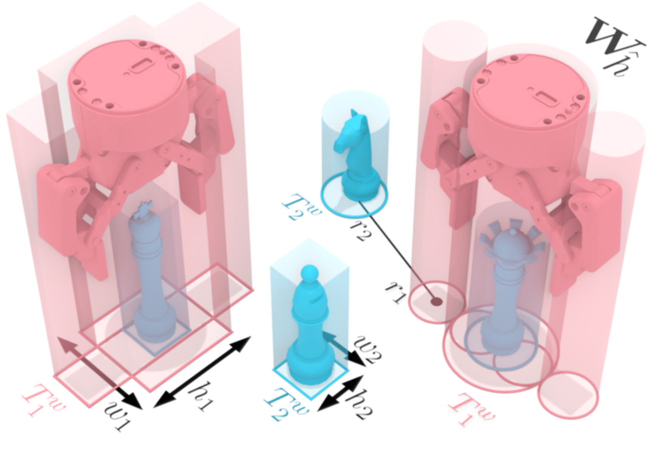


Fig. 2. Representation of projections of the robot hand and objects into the plane of manipulation for (left) AABB and (right) circles. Primitives $\mathbf{P}_o^t$ and $\mathbf{P}_r^t$ are shown as colored geometric primitives with their parameters, which are defined in Sec. V-B.

surface are grasped from above [25]. Compared to general TAMP, in OR interactions between the robot and the objects are simplified (e.g., top-down pick actions that are guaranteed to be feasible). Despite these assumptions, OR is known to be NP-hard [25]; complexity arises from constraints created by collisions in object start and goal poses. These constraints require finding intermediate object locations (necessary for non-monotone problems), which is non-trivial [17].

Insights from OR are used in our optimization-based grounding, which computes *optimal* intermediate placement locations in cluttered scenes. Note that in OR, it is usually assumed that all actions are feasible, and potential complex robot-object dependencies are ignored, such as the ones shown in this paper (e.g., Fig. 1). Our method does *not* assume action feasibility, and is able to figure out if intermediate locations are required based on feedback from our optimization-based grounder and motion planning. Recent works [17, 18] have analyzed OR in clutter, combining ideas from monotone solvers and motion planning. However, they still lack the capability of solving more general problems that TAMP methods such as ours provide.

III. Problem Definition

We focus on robot manipulation problems with a set of objects $\mathcal{O}$ in a workspace $\mathcal{W} \subseteq \mathbb{R}^3$. We use the Planning Domain Description Language (PDDL) [26] to define the discrete task domain—the task planner finds a *plan skeleton*, a sequence $\mathcal{A} = \{a_1, \ldots, a_K\}$ of action operators in the domain that transitions the system from the start to a goal.

We consider the class of problems of *planar manipulation*, where all objects are placed on a plane. Here, the robot's parallel jaw end-effector first reaches a pre-grasp pose, and then moves in a straight line in the workspace to attain the grasp pose. The *plane of manipulation* $\mathbf{W}_{\hat{h}}$ is defined by the normal vector $\hat{h}$ that represents the displacement of the robot's hand between pre-grasp and grasp poses. $\mathcal{M}_r : \mathcal{W} \rightarrow \mathbf{W}_{\hat{h}}$ and $\mathcal{M}_o : \mathcal{W} \rightarrow \mathbf{W}_{\hat{h}}$ are projections from points in the robot's hand and an object o respectively to $\mathbf{W}_{\hat{h}}$. Fig. 2 shows

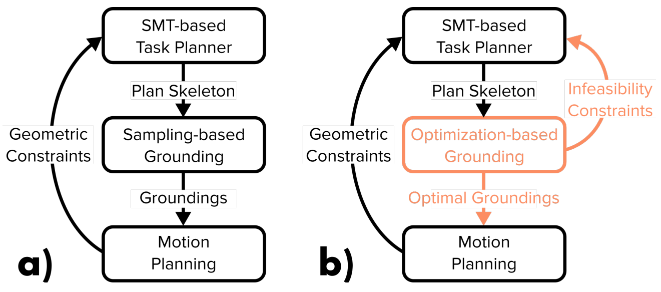


Fig. 3. **a)** Baseline TAMP framework [1]. **b)** Proposed TAMP framework for cluttered scenes. Our proposed TAMP framework optimization-based grounder provides additional feedback to the SMT-based task planner and optimal groundings to motion planning.

a visual representation of the these projections. Note that even if the motion to a pre-grasp pose exists, the corresponding motion from pre-grasp to grasp may not, i.e., due to geometric constraints between the objects and the robot.

We will assume that a trajectory between pre-grasp and grasp poses exists if the projections between the robot hand and objects not being manipulated onto $\mathbf{W}_{\hat{h}}$ do not intersect:

Assumption 1. *Let $\mathcal{X}_o^t$ and $\mathcal{X}_r^t$ be respectively the space occupied by object o and the robot at time t. Let $o^t = \{\bigcup \mathcal{M}_o(x_o^t) \mid \forall x_o^t \in \mathcal{X}_o^t\}, r^t = \{\bigcup \mathcal{M}_r(x_r^t) \mid \forall x_r^t \in \mathcal{X}_r^t\}$. If $o^t \cap r^t = \emptyset$, then there exists a collision-free trajectory from the configurations that achieve the pre-grasp pose to the grasp pose.*

The objects and robot's hand at step t are represented as sets of geometric primitives $\mathbf{P}_o^t$ and $\mathbf{P}_r^t$ in $\mathbf{W}_{\hat{h}}$:

$$\mathbf{P}_o^t = \left\{\bigcup p_{oi}^t \mid p_{oi}^t \subseteq \mathbf{W}_{\hat{h}}\right\}, \mathbf{P}_r^t = \left\{\bigcup p_{ri}^t \mid p_{ri}^t \subseteq \mathbf{W}_{\hat{h}}\right\}$$

Our choices for p_{oi}^t and p_{ri}^t are described in Sec. V-A. We will assume that $\mathbf{P}_o^t$ and $\mathbf{P}_r^t$ are conservative approximations of the objects and the robot hand at t:

Assumption 1a (Conservative Projection). *$\mathcal{M}_o(x_o^t) \in \mathbf{P}_o, \forall x_o^t \in \mathcal{X}_o^t$ and $\mathcal{M}_r(x_r^t) \in \mathbf{P}_r \forall x_r^t \in \mathcal{X}_r^t$.*

IV. Task and Motion Planning in Clutter

We provide an algorithmic framework to solve TAMP problems for manipulation planning in clutter. The framework, shown in Fig. 3b is based on [1]. It consists of an SMT solver for task planning that produces plan skeletons. A motion planner takes a plan skeleton that has been partially grounded and computes motions for action operators. Failure to find a motion plan introduces constraints that are used by the SMT solver to generate new plan skeletons.

Instead of using sampling or discretization of geometric variables to ground plan skeletons (Fig. 3a), our framework incorporates a novel optimization-based grounding method between the task planner and the motion planner (Fig. 3b). Our grounding method takes a plan skeleton and *jointly optimizes* geometric variables of ungrounded action operators – grasps and placements — before motion planning. Importantly, information about infeasible problems from the grounder is used to create new constraints that are added to the SMT solver. This mechanism allows our method to effectively consider geometric information at the task planner—which is critical for highly geometrically constrained and cluttered environments—*without explicitly incorporating additional geometric information at the symbolic level*. In this paper, we present instances of the grounding layer for classes of problems in planar manipulation planning that cover a wide variety of real-world applications. However, note that our framework is general and can be extended to other classes of problems.

V. Optimization-based Grounding

The input of our grounding layer is a plan skeleton $\mathcal{A}$, the initial pose of all objects in $\mathcal{W}$ and the goal pose of a subset $\mathcal{O}_{\text{goal}} \subseteq \mathcal{O}$ of the objects. Its output consists of optimal values for the robot's hand poses and intermediate placement locations for objects that are not in $\mathcal{O}_{\text{goal}}$ or a set of conflicting constraints if the model is infeasible (see Fig. 3b). The values of these geometric variables are critical to success in cluttered problems—choosing poor values early in a plan can create conflicts closer to the goal [16–18]. Note that these variables are computed only for each discretized action in the plan skeleton, i.e., when the robot grasps or releases an object. Geometric variables in-between actions (i.e., transit/transfer modes [27]) are handled by motion planning.

A. Robot and Objects Model

We propose using circles or axis-aligned bounding boxes (AABB) to represent $\mathbf{P}_o^t$ and $\mathbf{P}_r^t$. For a geometric primitive p_i^t, we represent its pose as $\mathcal{P}_i^t = (x_i^t, y_i^t, \theta_i^t) \in \text{SE}(2)$. The radii r_i (for circles) and width and height (w_i, h_i) (for AABBs) are inputs to the optimization.

B. Optimal Grounding

An optimal grounding over a plan skeleton is a solution to the following optimization problem:

$$\begin{aligned} \min_{\mathcal{P}^{1:K}} \quad & f(\mathcal{P}^{1:K}) \\ \text{st.} \quad & \mathbf{P}_{Ob(a_t)}^t \cap \mathbf{P}_k^t = \emptyset, \forall k \in \mathcal{O} \setminus Ob(a_t), \forall a_t \in \mathcal{A} && \text{(1a)} \\ & \mathbf{P}_r^t \cap \mathbf{P}_k^t = \emptyset, \forall k \in \mathcal{O} \setminus Ob(a_t), \forall a_t \in \mathcal{A} && \text{(1b)} \\ & \mathbf{P}_o^0 = \mathbb{S}_o, \mathbf{P}_u^K = \mathbb{G}_u, \forall o \in \mathcal{O}, \forall u \in \mathcal{O}_{\text{goal}} && \text{(1c)} \\ & \mathbf{P}_o^{t+1} = \mathbf{P}_o^t, \forall o \in \mathcal{O} \setminus Ob(a_t) && \text{(1d)} \end{aligned}$$

where $Ob(a_t)$ corresponds to the object being manipulated at action a_t, $\mathcal{P}^t = \left\{\mathcal{P}_{r0}^t, \ldots, \mathcal{P}_{r|\mathbf{P}_r^t|}^t, \mathcal{P}_{o0}^t, \ldots, \mathcal{P}_{o|\mathbf{P}_o^t|}^t\right\}$ is the stacked vector of poses for every object and robot primitive at t, $\mathbb{S}_o$ and $\mathbb{G}_u$ are start/goal poses for objects o and u and f is a user-defined convex function. In our experiments (see Sec. VI), f is the sum of quadratic displacements over objects that encourages minimal end-effector motion, but this choice can change depending on the problem at hand.

Eq. (1a) and Eq. (1b) represent no intersection between pairs of geometric primitives in $\mathbf{W}_{\hat{h}}$. Eq. (1a) forces the primitives of the manipulated object at t to not intersect with the primitives of other objects at that timestep. Similarly, Eq. (1b) forces the primitives of the robot to not intersect with the primitives of the objects, except for the one being manipulated. Eq. (1c) models the start and goal configurations for object primitives and Eq. (1d) expresses that primitives of an object will maintain their previous poses unless the object is manipulated by the action at that timestep.

A solution to the *optimal grounding* formulation above is a vector of the robot's hand and object poses primitives for all steps in $\mathcal{A}$ with the minimum value for the objective function f for which trajectories between pre-grasp and grasp poses are guaranteed to exist.

C. Primitive Parameterizations

Next, we show specific instances of the pairwise non-intersection constraints Eq. (1a) and Eq. (1b) for two object models: AABBs and circles. We note that the following formulations both provide *global optimality guarantees*, that is, if a solution is found, it is guaranteed to be the optimal solution. Moreover, if the optimizer declares infeasibility, it is because there exists subsets of the constraints that can not be jointly satisfied. This fact is exploited by our solver, as described in Sec. V-D. Note that other object models are possible—Eq. (1) is agnostic to the specific implementation.

1) AABB: In this model, primitives in $\mathbf{P}_0^t$ and $\mathbf{P}_r^t$ are AABBs with parameters (x, y, w, h, θ). The robot can grasp objects using hand orientations that are aligned with the axis of the supporting plane, i.e., $\theta \in \{0, \pi/2\}$. Two AABBs will not intersect if at least one of the following constraints holds (see Fig. 2):

$$\begin{aligned} C_1 &: x_1 - z_1^{(1)}/2 \geq x_2 + z_2^{(1)}/2, \\ C_2 &: x_2 - z_2^{(1)}/2 \geq x_1 + z_1^{(1)}/2, \\ C_3 &: y_1 - z_1^{(2)}/2 \geq y_2 + z_2^{(2)}/2, \\ C_4 &: y_2 - z_2^{(2)}/2 \geq y_1 + z_1^{(2)}/2, \\ z_i^{(1)} &= (w_i(1 - b_{\theta i}) + h_i b_{\theta i}), \\ z_i^{(2)} &= (h_i(1 - b_{\theta i}) + w_i b_{\theta i}) \end{aligned} \tag{2}$$

where $b_{\theta i} \in \{0, 1\}$ indicates the orientation of the i-th AABB (0 or $\pi/2$ respectively). In consequence, non-intersection constraints can be expressed for each pair of AABB primitives as the disjunctive constraint $\bigvee_{i=1}^{4} C_i$, which in turn can be expressed as a conjunction of constraints with additional binary variables [28]. The overall formulation becomes a mixed-integer program (MIP), since it contains continuous variables (for placement locations), binary variables (for the grasping choice) and linear constraints. Mathematical programs with these characteristics can be efficiently solved using off-the-shelf solvers [29].

2) Circles: In this model all primitives are circles parameterized with (x, y, r, θ). The general form of the non-intersection constraints can be written as:

$$\|T_1^w(x_1, y_1, \theta_1)c_1 - T_2^w(x_2, y_2, \theta_2)c_2\|^2 \geq (r_1 + r_2)^2 \tag{3}$$

where c_1, c_2 are the centers of the circles in their local reference frame and T_i^w are rigid transformations from the local frame of primitive i to $\mathbf{W}_{\hat{h}}$ (see Fig. 2). We also consider hand orientations that are aligned with the axis of the manipulation plane, i.e., $\theta_1, \theta_2 \in \{0, \pi/2\}$. In that case, Eq. (3) can be expressed in general form as $g(x) \leq 0$, where $g(x)$ is a non-convex quadratic function with both integer and continuous variables, where:

$$\begin{aligned} T^w(x, y, \theta)c = &(c_x + x + c_y + y)(1 - b_\theta) \\ &+ (x + c_x + y - c_y)b_\theta \end{aligned} \tag{4}$$

where $b_\theta \in \{0, 1\}$ indicates whether θ is 0 or $\pi/2$. When using this model, the problem is a non-convex mixed-integer quadratically constrained program (MIQCP). Although non-convex, modern optimization solvers can efficiently solve these problems to global optimality using bilinear programming and branch-cut techniques [29]. Although this model permits continuous hand orientations, we have restricted it to a discrete set to provide global optimality guarantees. A formulation with continuous orientations may require expressing the problem as a nonlinear program (NLP), where these guarantees may not hold.

D. Providing Feedback to the Task Planner

When the grounding model is infeasible, information about this infeasibility can be used to guide the task planner to produce plan skeletons that are more likely to be feasible. Due to the optimality guarantees of our grounding layer, this can be achieved by identifying potential reasons of infeasibility in the underlying optimization model, i.e., by identifying conflicting constraints at the converged solution. We do this by computing an Irreducible Inconsistent Subsystem (IIS) [30, 31], a subset of inconsistent constraints that becomes feasible when one is removed from the set. Within the IIS, constraints Eq. (1a) and Eq. (1b) are identified and used to construct symbolic expressions to block the actions and groundings that originated them. In particular, we identify the timestep t, the action operator a_t, the conflicting objects $Ob(a_t)$ and k to create the following *symbolic* constraint:

$$\bigwedge_{s=1}^{K} \left(l_k^{[s]} \rightarrow \neg a_t^{[s]} \right)$$

where $l_k^{[s]}$ is a proposition that evaluates a predicate related to object k, e.g., whether the object is at a given location or if the object is on top of another object. When added to the task planner, these expressions will block operator a_t at every timestep in the plan skeleton where the proposition holds. For the examples in this paper, some of these constraints may have the following intuition: "do not pick up object A from location X while object B is in location Y" or "do not stack object A into object B when object C is at location X", etc. Modern optimization solvers [29] can efficiently compute an IIS from an infeasible formulation, e.g., based on [32].

VI. Experiments

We implement our method with Z3 [8] as the SMT solver, RRT-Connect [13] in OMPL [33] through Robowflex [34] and DART [35] for motion planning, and Gurobi 9.5 [29] for our optimization-based grounder.

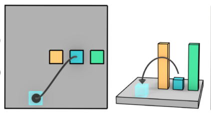
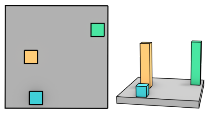
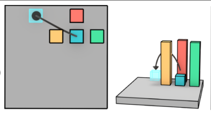
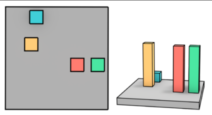
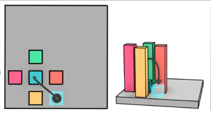
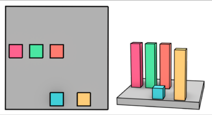
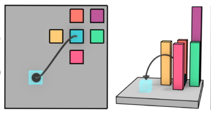
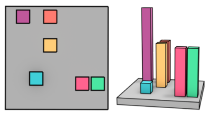
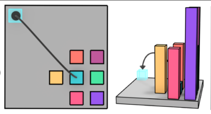
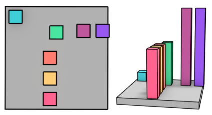
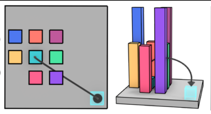
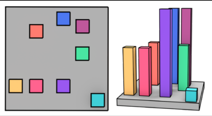
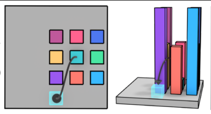
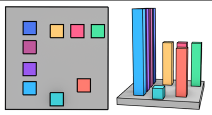
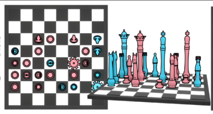
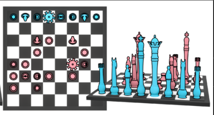
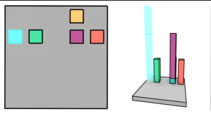
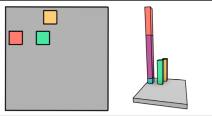
Problem				Planning Statistics					Plan Statistics		
	Example Start	Example Goal	Method	Total (s)	TP	GD	IIS	MP	Makespan	EE Disp. (m)	Success
OP-3			SMP1	9.31±38.46	87%			13%	7.76±2.01	1.89±0.80	**100%**
			SMP2	9.65±37.31	86%			14%	7.52±2.00	1.80±0.56	**100%**
			OPT	**0.19±0.07**	79%	9%		13%	**6.00±0**	1.37±0.12	**100%**
			OPT-IIS	0.21±0.01	60%	6%	29%	5%	**6.00±0**	**1.37±0.10**	**100%**
OP-4			SMP1	24.05±45.25	92%			8%	10.08±2.20	2.29±0.81	98%
			SMP2	23.01±37.67	91%			9%	10.24±2.03	2.23±0.52	98%
			OPT	0.81±0.19	84%	8%		8%	**8.00±0**	1.55±0.55	**100%**
			OPT-IIS	**0.71±0.02**	63%	5%	29%	2%	**8.00±0**	**1.52±0.24**	**100%**
OP-5			SMP1	253.14±232.17	79%			21%	10.62±0.94	3.46±3.77	58%
			SMP2	256.11±194.38	74%			26%	10.73±0.98	2.56±0.84	44%
			OPT	6.29±3.63	92%	4%		4%	**10.00±0**	2.02±2.51	**100%**
			OPT-IIS	**2.08±0.04**	72%	5%	23%	1%	**10.00±0**	**1.74±0.48**	**100%**
OP-6			SMP1	-	-	-	-	-	-	-	0%
			SMP2	-	-	-	-	-	-	-	0%
			OPT	124.49±100.72	93%	2%		6%	**12.00±0**	**2.33±1.10**	96%
			OPT-IIS	**6.72±0.97**	83%	3%	15%	0%	**12.00±0**	2.34±1.92	**100%**
OP-7			SMP1	-	-	-	-	-	-	-	0%
			SMP2	-	-	-	-	-	-	-	0%
			OPT	301.80±299.32	95%	1%		4%	**14.00±0**	4.60±2.41	4%
			OPT-IIS	**19.98±3.75**	89%	2%	9%	0%	**14.00±0**	**4.13±4.58**	**100%**
OP-8			SMP1	-	-	-	-	-	-	-	0%
			SMP2	-	-	-	-	-	-	-	0%
			OPT	-	-	-	-	-	-	-	0%
			OPT-IIS	**62.90±13.88**	93%	1%	6%	0%	**16.00±0**	**4.90±4.65**	**98%**
OP-9			SMP1	-	-	-	-	-	-	-	0%
			SMP2	-	-	-	-	-	-	-	0%
			OPT	-	-	-	-	-	-	-	0%
			OPT-IIS	**144.43±41.29**	95%	1%	3%	0%	**18.00±0**	**6.22±4.95**	**94%**
Chess			SMP1	-	-	-	-	-	-	-	0%
			SMP2	-	-	-	-	-	-	-	0%
			OPT	490.24±198.55	100%	0%		0%	**16.00±0**	**5.44±0.83**	18%
			OPT-IIS	**456.44±206.40**	98%	0%	2%	0%	17.04±1.09	5.49±1.28	**96%**
Tower			SMP1	-	-	-	-	-	-	-	0%
			SMP2	-	-	-	-	-	-	-	0%
			OPT	-	-	-	-	-	-	-	0%
			OPT-IIS	**11.51±2.37**	93%	1%	6%	0%	**14.00±0**	**3.60±2.17**	**96%**

Table I. Summary of experiments. An example start and goal configuration of each problem are shown on the left. Each row represents 50 randomized trials of each method with a 1000 second timeout, with mean $\pm$ standard deviation of successful runs. In *OP-X*, only the goal location of the light blue cube is specified (indicated by the arrow and transparent cube in the start column). In *Chess*, the goal is to move all 8 blue pieces pieces to their starting positions. *Tower* requires three specific blocks to be stacked at a goal location (highlighted in light blue). In planning statistics, the total time spent in seconds is given, along with the mean percentage spent in task planning (TP), grounding (GD), IIS computation (IIS), and motion planning (MP). In plan statistics, the plan's makespan (total number of actions) and total end-effector displacement are given. The success rate of each method is on the right.

A. Simulated Environments

We have created highly cluttered environments where the robot (a UR5 adjacent to the small table, shown in Fig. 1) is tasked with achieving a high-level goal, described in PDDL. All the environments require reasoning over complex object-object and object-robot geometric relationships for axis-aligned top-down grasps, making the manipulation problems challenging. For all environments, we create a set of 50 different problems by randomly sampling the start and goal states. All experiments have a total TAMP planning timeout of 1000 seconds. Results are summarized in Table I.

a) Obstructed Pick X (OP-X): The robot needs to move a short object surrounded by $X-1$ taller objects to a desired goal location. The close proximity and difference in height makes grasping the short object impossible when next to the taller objects. When moving the taller objects away, placement locations need to be computed, which is expected to be harder for higher X, due to less available space. These experiments are modeled using the AABB model described in Eq. (2) and demonstrate scalability of our method, as difficulty increases with the number of objects.

b) Chessboard (Chess): The robot must setup the 8 blue pieces in their starting position (Fig. 1). The initial configuration of both the red and blue pieces is randomly sampled. Short pieces can not be grasped if next to a taller piece, e.g., a king or a queen. Similarly, shorter pieces (e.g., bishops) can not be placed at their goal locations if tall pieces have already been placed. This experiment is modeled with circles, as described in Eq. (4), and demonstrates long-horizon task plans that use general mesh representations for objects.

c) Tower Assembling (Tower): The robot assembles a tower from a subset of objects on the table. Each object is distinct; chosen objects must be stacked in a specific order at the goal. The objects have different heights, making manipulation of the shorter objects next to taller objects infeasible. Placement locations need to be grounded while considering both geometric constraints and task-level constraints (e.g., object A is on object B, and if the goal is to stack them in the same order, A has to be placed at a buffer to move B to the goal). This problem is modeled with AABBs, and includes challenges similar to *OP-X*, with increased difficulty in both the task and motion planning domains.

B. Planning Domain

The task planning domain contains symbols for the start, goal, and potential intermediate location of each object. We define action operators that transfer an object from the start (PICK), transfer the object to the goal (PLACE), as well as specialized operators to use the intermediate location (PLACE-BUFFER and PICK-BUFFER). For the *Tower* problem, there are also actions to stack a block on top of another block (STACK) and to remove a block from the top of a block (UNSTACK). When a goal or intermediate location is used in a plan skeleton, it becomes a variable that is decided by the grounding method. This model is more efficient than incorporating symbols for every location, region, grasp pose, or other geometric information [6].

C. Results

We compare the performance of our framework against variations of [1], described below. These methods can prune multiple task plans by blocking state-action pairs for all the steps in the horizon when motion planning fails. Additionally, we compare our full framework with a variation that optimizes over plan skeletons but that does not implement task planner feedback:

- *SMP1*: A method similar to [1] that samples grasp poses and intermediate locations for actions that require them.
- *SMP2*: Similar to *SMP1*, but if the sampled grasp pose fails, a new attempt is made using a different grasp pose.
- *OPT*: A variation of our optimization-based grounding method which does *not* give any feedback to the task planner. When a plan skeleton is found infeasible by the grounder, this method falls back to sampling.
- *OPT-IIS*: This is the full implementation of our method as described in Sec. IV, including feedback from the optimization model to the task planner.

To evaluate the methods we compute the total time taken by the planner and we show the percentage of that time used for task planning (TP), grounding (GD), IIS computation (IIS) and motion planning (MP). We also compute statistics about the plan, such as the length of the found task plan (*Makespan*), the displacement of the robot's end effector (*EE Disp.*) and the percentage of problems solved (*Success*).

Table I summarizes results with a visualization of an example start and goal for each environment. We first focus on the *OP-X* set of experiments and compare the performance of the baselines with our proposed framework when increasing the difficulty of the problem. In all cases, task planning time dominates over all the other steps in the pipeline. However, it is noteworthy that the process of computing the IIS becomes more expensive (in absolute time) for problems with more pairwise collision constraints (e.g., *OP-8*, *OP-9*). These results show the benefits of using our TAMP framework for environments that are highly cluttered.

Methods based on sampling can solve almost all the problems in *OP-3* and *OP-4* and around half of the problems in *OP-5*, but fail to solve any problem in the hardest instances. Note that when *SMP1–2* find a solution, they take orders of magnitude more time than *OPT-IIS*, most of which is spent task planning. This happens because it is often the case that the sampled grasp configurations and intermediate locations are not feasible due to the clutter, forcing the planner to block actions that could have been feasible if appropriate values had been attempted. *OPT* is capable of solving *OP-6*, but only a few instances of *OP-7*, and none of *OP-8* and *OP-9*. *OPT-IIS* solves almost all instances of the *OP-X* problems, demonstrating the importance of both the optimization and the feedback. Note *OPT-IIS* finds the plan with the minimum makespan and lowest end-effector displacement.

Chess can be solved by *OPT* and *OPT-IIS*, but not by the baselines that only use sampling. Baseline methods fail to solve this problem within timeout, as task planning is very expensive and it is challenging to discover feasible sequences of grasps for the pieces given the complex geometric constraints. We note the high success rate of *OPT-IIS*, emphasizing the strength of using feedback to guide the search.

Tower is also shown to be highly challenging for the baseline methods. Here, complex relationships between objects prevent grasping and stacking of shorter objects next to taller ones. For example, poorly chosen buffer locations for unstacked objects will interfere with the goal location of the tower. *OPT-IIS* can efficiently discover such relations through feedback, while other methods completely fail.

VII. Concluding Remarks

We have proposed a novel optimization-based grounding approach that enables TAMP methods to solve complex, cluttered manipulation planning problems. Between the task planner and motion planner, our novel grounding layer finds optimal values for action parameters in a candidate plan skeleton, and if no feasible grounding is possible, also provides feedback to the task planner to guide search towards feasible actions. Our grounding approach handles planar manipulation, an important class of problems in TAMP, and provides global optimality guarantees—in future work, we plan to investigate broader classes of problems and object approximations. Further investigation is also merited in the use of infeasibility information to guide task planning, and additionally how optimality guarantees can be used to provide information to guide planning as well. We also plan to extend our framework to cases where Asm. 1 does not hold, e.g., obstacles outside the plane of manipulation that prevent the motion planner from finding a solution.

References

[1] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki. “An incremental constraint-based framework for task and motion planning”. In: *Int. J. of Robotics Research* 37.10 (2018), pp. 1134–1151.

[2] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. “FFRob: Leveraging symbolic planning for efficient task and motion planning”. In: *Int. J. of Robotics Research* 37.1 (2018), pp. 104–136.

[3] 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: *IEEE Int. Conf. Robot. Autom.* 2014, pp. 639–646.

[4] M. Toussaint. “Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning”. In: *Proceedings of the 24th International Conference on Artificial Intelligence*. IJCAI’15. Buenos Aires, Argentina: AAAI Press, 2015, pp. 1930–1936.

[5] Z. Kingston and L. E. Kavraki. “Scaling Multimodal Planning: Using Experience and Informing Discrete Search”. In: *IEEE Transactions on Robotics* (Aug. 2022), pp. 1–19.

[6] T. Pan, A. M. Wells, R. Shome, and L. E. Kavraki. “A General Task and Motion Planning Framework For Multiple Manipulators”. In: *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.* 2021, pp. 3168–3174.

[7] J. Ortiz-Haro, E. Karpas, M. Katz, and M. Toussaint. “A Conflict-Driven Interface Between Symbolic Planning and Nonlinear Constraint Solving”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 10518–10525.

[8] L. de Moura and N. Bjørner. “Z3: An Efficient SMT Solver”. In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by C. R. Ramakrishnan and J. Rehof. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 337–340.

[9] J. Hoffmann. “FF: The fast-forward planning system”. In: *AI magazine* 22.3 (2001), pp. 57–57.

[10] J. Rintanen. “Madagascar: Scalable planning with SAT”. In: *Proceedings of the 8th International Planning Competition (IPC-2014)* 21 (2014), pp. 1–5.

[11] L. E. Kavraki, P. Svestka, J. C.-. Latombe, and M. H. Overmars. “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”. In: *IEEE Transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

[12] S. M. Lavalle, J. J. Kuffner, and Jr. “Rapidly-Exploring Random Trees: Progress and Prospects”. In: *Algorithmic and Computational Robotics: New Directions*. 2000, pp. 293–308.

[13] J. Kuffner and S. LaValle. “RRT-connect: An efficient approach to single-query path planning”. In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. Vol. 2. 2000, 995–1001 vol.2.

[14] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour. “Manipulation Planning Among Movable Obstacles”. In: *IEEE Int. Conf. Robot. Autom.* 2007, pp. 3327–3332.

[15] M. Stilman and J. Kuffner. “Planning Among Movable Obstacles with Artificial Constraints”. In: *Int. J. of Robotics Research* 27.11-12 (2008), pp. 1295–1307.

[16] J. A. Haustein, K. Hang, J. Stork, and D. Kragic. “Object Placement Planning and optimization for Robot Manipulators”. In: *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.* 2019, pp. 7417–7424.

[17] K. Gao, D. Lau, B. Huang, K. E. Bekris, and J. Yu. “Fast High-Quality Tabletop Rearrangement in Bounded Workspace”. In: *IEEE Int. Conf. Robot. Autom.* 2022.

[18] R. Wang, Y. Miao, and K. E. Bekris. “Efficient and High-quality Prehensile Rearrangement in Cluttered and Confined Spaces”. In: *IEEE Int. Conf. Robot. Autom.* 2022.

[19] A. Kimmel, R. Shome, and K. Bekris. “Anytime motion planning for prehensile manipulation in dense clutter”. In: *Advanced Robotics* 33.22 (2019), pp. 1175–1193.

[20] M. Toussaint, K. R. Allen, A. Smit, and J. B. Tenenbaum. “Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning”. In: *Robotics: Science and Syst.* 2018.

[21] A. M. Wells, N. T. Dantam, A. Shrivastava, and L. E. Kavraki. “Learning Feasibility for Task and Motion Planning in Tabletop Environments”. In: *IEEE Robotics and Automation Letters* 4.2 (Apr. 2019), pp. 1255–1262.

[22] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. “Sampling-based methods for factored task and motion planning”. In: *The International Journal of Robotics Research* 37.13-14 (2018), pp. 1796–1825.

[23] J. Ortiz-Haro, V. N. Hartmann, O. S. Oguz, and M. Toussaint. “Learning efficient constraint graph sampling for robotic sequential manipulation”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 4606–4612.

[24] J. Ichnowski, M. Danielczuk, J. Xu, V. Satish, and K. Goldberg. “GOMP: Grasp-Optimized Motion Planning for Bin Picking”. In: *IEEE Int. Conf. Robot. Autom.* 2020, pp. 5270–5277.

[25] S. D. Han, N. M. Stiffler, A. Krontiris, K. E. Bekris, and J. Yu. “Complexity Results and Fast Methods for Optimal Tabletop Rearrangement with Overhand Grasps”. In: *The International Journal of Robotics Research* 37.13-14 (2018), pp. 1775–1795.

[26] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. *PDDL — The Planning Domain Definition Language*. Tech. rep. Yale Center for Computational Vision and Control, 1998.

[27] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. “Integrated Task and Motion Planning”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 4.1 (2021), pp. 265–293.

[28] L. Blackmore, M. Ono, and B. C. Williams. “Chance-Constrained Optimal Path Planning With Obstacles”. In: *IEEE Transactions on Robotics* 27.6 (2011), pp. 1080–1094.

[29] L. Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2022.

[30] J. van Loon. “Irreducibly Inconsistent Systems of Linear Inequalites”. In: *European Journal of Operational Research* 8.3 (1981), pp. 283–288.

[31] J. W. Chinneck. *Feasibility and Infeasibility in Optimization:: Algorithms and Computational Methods*. Vol. 118. Springer Science & Business Media, 2007.

[32] J. Gleeson and J. Ryan. “Irreducibly inconsistent systems of linear inequalities”. In: *INFORMS journal on Computing* 8.3 (1981), pp. 283–288.

[33] I. A. Sucan, M. Moll, and L. E. Kavraki. “The Open Motion Planning Library”. In: *IEEE Robot. Autom. Magazine* 19.4 (2012), pp. 72–82.

[34] Z. Kingston and L. E. Kavraki. “Robowflex: Robot Motion Planning with MoveIt Made Easy”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, May 2022.

[35] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu. “DART: Dynamic Animation and Robotics Toolkit”. In: *Journal of Open Source Software* 3.22 (2018), p. 500.