SyDeBO: Symbolic-decision-embedded bilevel optimization for long-horizon manipulation in dynamic environments

ZHIGEN ZHAO1†, ZIYI ZHOU1†, MICHAEL PARK1, 2 AND YE ZHAO1, (IEEE Member)

1Laboratory for Intelligent Decision and Autonomous Robots, George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30313, USA. {zhigen.zhao, zhouziyi, michael.j.park, yzhao301} @gatech.edu
2Georgia Tech Research Institute, Atlanta, GA 30318, USA
†the first two authors are equally contributed.

ABSTRACT This study proposes a Task and Motion Planning (TAMP) method with symbolic decisions embedded in a bilevel optimization. This TAMP method exploits the discrete structure of sequential manipulation for long-horizon and versatile tasks in dynamic environments. At the symbolic planning level, we propose a scalable decision-making method for long-horizon manipulation tasks using the Planning Domain Definition Language (PDDL) with causal graph decomposition. At the motion planning level, we devise a trajectory optimization (TO) approach based on the Differential Dynamic Programming (DDP) and Alternating Direction Method of Multipliers (ADMM), suitable for solving constrained, large-scale nonlinear optimization in a distributed manner. Distinct from conventional geometric motion planners, our approach generates highly dynamic manipulation motions by incorporating the full robot and object dynamics. Furthermore, in lieu of a hierarchical planning approach, we solve a holistically integrated bilevel optimization problem involving costs from both the low-level TO and the high-level search. Simulation and experimental results demonstrate dynamic manipulation for long-horizon object sorting tasks in clutter and on a moving conveyor belt.

INDEX TERMS Task and motion planning, trajectory optimization

I. INTRODUCTION

Long-horizon robot manipulation such as those observed in industrial assembly and logistics often involves hard-coded and repetitive motions. This defect severely limits manipulation task variety and flexibility. Consider a sorting task of objects in clutter and on a conveyor belt shown in Fig. 1, where multiple object manipulation sequences are possible, and manipulator’s reach to an object might be blocked by another object. The computational burden to plan for such a long-horizon sequential manipulation scenario introduces a significant challenge in robot task and motion planning. Artificial intelligence (AI) planning approaches have made significant progress in handling symbolic plan search and tasks constraints [1], [2]. However, such traditional AI planning methods often disregard the low-level physical constraints and dynamics when evaluating the cost and the feasibility of a plan, which poses a challenge to robotics problems often involving complex dynamics and contact physics.

Recent attention has been drawn to dynamic manipulation that involves versatile interactions with cluttered environments or complex objects. For instance, how can a robot arm with a two-parallel-jaw gripper manipulate an oversized package by a pushing action or throw a parcel into an unreachable cart? State-of-the-art trajectory optimization (TO) methods for contact-rich robotic systems often incorporate intrinsically hybrid contact dynamics either in a hierarchical [3], [4] or a contact-implicit [5] fashion, where the TO is formulated either with or without knowing the contact mode...
sequence a priori. However, existing TO has a limitation in designing robot motions associated with a combinatorics of local minima. This results from the challenge of designing costs and constraints for a highly nonlinear optimization problem and the difficulty of algorithm convergence over a long trajectory duration. As far as we know, none of the above TO approaches reason about high-level task planning.

Task and motion planning (TAMP) aims to simultaneously solve for sequential task composition and underlying robot motion. Recent works on TAMP use an optimization-based approach to solve for an action skeleton while meeting dynamics constraints from the robot manipulators and intricate contact events [6], [7]. Following this line of research, our study combines symbolic search with a full-model-based optimization under the framework of bilevel TO. We use PDDL to design expressive logic formulas for complex grasping tasks and physical interaction representations. The discrete states and actions defined in PDDL will be encoded as constraints in the bilevel TO as shown in Fig. 2. To alleviate the computational burden of solving multiple TOs within a symbolic search, our study uses causal graph to decompose the long-horizon planning problem into multiple sub-tasks and sequentially solve a combined symbolic planning and TO as bilevel optimization via multi-stage search. A similar idea to simplifying problems through decomposition has been used in [8] and [9], but with different decomposition approaches.

To solve the low-level TO, the Alternating Direction Method of Multipliers (ADMM) approach [10] is employed to provide a general framework capable of handling various constraints [11], [12] including contact constraints, collision avoidance constraints, and manipulation task constraints. This approach decomposes the full TO into three types of ADMM blocks, each of which is updated iteratively. The synthesized high-level symbolic planner will govern the activity of different ADMM blocks in two aspects. On one hand, the symbolic planner will select the ADMM blocks under each block type according to different action candidates, so that a specific low-level TO can be constructed. On the other hand, once each type of ADMM block is determined, these ADMM blocks will be activated or deactivated during the multi-stage search to perform either a full or partial ADMM update. The solution of the first-stage search provides a primitive warm-start to the second-stage so that the full optimization can be solved computationally efficiently.

The proposed symbolic-decision-embedded bilevel optimization (SyDeBO) takes a step towards unifying the high-level task planning and low-level dynamics-consistent trajectory optimization into a coherent TAMP framework for long-horizon manipulation. The contributions of this study lie in the following:

- Propose a causal graph method at the discrete planning domain to identify and decompose sub-tasks. The separation of sub-tasks simplifies the entire problem by limiting the actions and predicates to a relevant subset.
- Devise an ADMM-based distributed optimization to incorporate various sets of constraints, which are enforced by the symbolic planner. This distributed structure enables a flexible switching mechanism for activating and deactivating ADMM blocks, well suited for being integrated with a discrete symbolic planner.
- Evaluate the feedback control performance of our holistic bilevel optimization in both dynamic simulation and hardware experiments for long-horizon manipulation in clutter and a moving conveyor belt environment.

The organization of this paper is as follows: Sec. II surveys the related works on task and motion planning and trajectory optimization for manipulation; Sec. III formulates TAMP for sequential manipulation as a bilevel optimization problem, and describes a causal graph decomposition method and a symbolic multi-stage search algorithm to solve the bilevel optimization; Sec. IV provides more details for the low-level distributed trajectory optimization framework; Sec. V presents the experimental results in both simulation and hardware; Sec. VI summarizes the proposed TAMP framework and discusses the future directions.

II. RELATED WORK

Hybrid Planning in Manipulation: Hybrid Differential Dynamic Programming [13], [14] aims to solve a hybrid optimal control problem combining discrete actions and continuous control inputs. The work in [13] optimized continuous mixtures of discrete actions. The authors in [14] used an exhaustive search over all hybrid possibilities with the cost computed by an input-constrained DDP. However, both of these two works are limited to simple hybrid modes such as discrete contact locations, instead of expressive symbolic actions as the ones proposed in our work. Moreover, the input-constrained DDP doesn’t support any state inequality constraints inside the optimization. Stouraitis [9] proposes a bilevel structure to combine continuous optimization with discrete graph-search for collaborative manipulation. This bilevel formulation not only mitigates the poor convergence of solving the whole optimization problem, but also holistically reasons about variables from each level.

Task and Motion Planning: TAMP for dynamic robot manipulation has become a powerful method to explicitly define symbolic tasks and enable a diverse set of manipulation motions. Woodruff and Lynch proposed a hierarchical planning approach for hybrid manipulation that defined a sequence of motion primitives a priori [15]. Garrett et. al. proposed a sampling-based method that incorporated sampling procedures as stream symbols in PDDL [16], [17]. However, the sampling-based methods focus on identifying feasible solutions rather than optimizing underlying trajectory cost. The work in [17] uses causal-graph-based FastDownward solver [2] as subroutine. In our work, evaluating trajectory costs for each symbolic action requires prior knowledge of robot poses from the previous action sequence. This requirement limits our high-level task planning to forward state-space search, which makes it challenging to adopt refinement-based
task planners, such as [2]. Along another line of research, Toussaint proposed the logic-geometric programming (LGP) that embedded the high-level logic representation into the low-level optimization [6], [18]. A more recent work in [7] extended LGP to provide general physical reasoning. The work in [19] adapted this task and motion planning method to object-centric manipulation in a dynamic environment.

Distributed Trajectory Optimization: ADMM has gained increasing attention in the robotics arena for solving parallel, large-scale motion planning problems. As a special case of Douglas-Rachford splitting methods [20], the classical ADMM was formulated to solve an optimization where the cost function is separable into two sub-blocks along with a coupled linear constraint [10]. ADMM has been further explored in [21] to solve constrained optimization problems with box and cone constraints. Although formally provable convergence for nonconvex problems can only be guaranteed under specific assumptions [22], ADMM is powerful in practice and has been widely studied for nonlinear robotic problems [11], [12]. More specifically, our previous work [12], [23] proposed a framework embedding DDP as a sub-block to solve rigid body dynamics. Inspired by these works above, this study formulates a bilevel optimization that combines an ADMM-based trajectory optimization with a high-level multi-stage search.

Bilevel Optimization for Manipulation: Our study is inspired by Toussaint’s LGP [6], [18] and Stouraitis’s bilevel optimization [9]. LGP uses multi-bounds heuristics to efficiently handle symbolic search and bound generalization to prune infeasible branches. However, in dynamical environments, the spontaneous changes of the environment could make bound generalization invalid. Our work focuses on the scalability of the algorithm using task decomposition. Since it does not directly prune the state-space search tree, the proposed task decomposition could be integrated with other TAMP frameworks, such as [18], for both improved scalability and efficient state-space tree search. In the low-level TO, instead of using interior point [9] or augmented Lagrangian methods [6], our ADMM-based solver exploits the optimization structure and splits the full-optimization problem into multiple ADMM blocks, each of which can be solved independently. This distributed structure fits well into the high-level discrete search.

III. PROBLEM FORMULATION

A. SYMBOLIC ACTION REPRESENTATION

In this paper we examine sequential manipulation planning for object sorting tasks, where a robot manipulates various objects into their respective goal locations. The symbolic predicates define the discrete state of the planning domain and impose constraints in DDP-ADMM trajectory optimization. For example, the (free X) predicates represents whether the end-effector of robot X is empty and free to grasp new object. The (on Y Z) predicates constraints object Y to be on table Z. (at X Y) indicates the end-effector of robot X is in a ready position to grasp the object Y. Additionally, (unobstructed A B) indicates whether object B is obstructing the manipulator’s reach to object A.

Our symbolic manipulation actions include Grasp (X, Y, Z), Move (X, Z), Release (X, Y, Z), Push (X, Y, Z), and Throw (X, Y, Z). For instance, Grasp (X, Y, Z) action allows a robot X to grasp an object Y on a surface Z, either from the top or the side. The preconditions of this grasping action are threefold: (free X), (at X Y), and (on Y Z). The robot holds the target object as the outcome of this action.

- Grasp (X, Y, Z) action allows a robot X to grasp an object Y on a surface Z, either from the top or the side. The preconditions of this grasping action are threefold: (i) the end-effector is not holding any object, (ii) the end-effector is in a ready position to grasp the object Y, and (iii) the object Y is on the surface Z. The robot holds the target object from the top or the side as the outcome of this action.

- Move (X, Z) action allows a robot X to move to a location Z. The preconditions of the move actions are independent from the grasp and the release actions,
meaning that the gripper may or may not be holding an object. The robot X is located at the position Z as the outcome of this action.

- **Release**(X, Y, Z) action allows a robot X to place a object Y at a location Z. The preconditions of this action include that the end-effector is moved to the drop-off position while holding the object. As the outcome, the robot no longer holds the object, and the object is placed at the location Z.

- **Push**(X, Y, Z) action allows a robot X to push an object Y to a location Z. The preconditions of this action are that the end-effector is moved to a ready position for pushing without holding any objects. The effect of the action is that the object Y is placed at the location Z.

- **Throw**(X, Y, Z) action allows a robot X to throw an object Y to a location Z. The preconditions are that the end-effector is holding object Y, and that the robot is moved to a position ready for throwing. The effect of the action is that object Y is at location Z. After the gripper release, the object follows a free fall motion.

**B. CAUSAL GRAPH TASK DECOMPOSITION**

To reduce the size of the symbolic search space, we decompose the entire symbolic goal into sub-goals to be achieved and maintained sequentially. Each sub-goal contains a disjoint subset of the goal predicates. We define a sub-task to be a sub-goal and the objects, predicates, and actions that need to be considered for the sub-goal. To identify the sub-tasks, we analyze the object dependencies using the causal graph of the symbolic planning domain. The causal graph is constructed similarly to the definition [2].

Constructing the causal graph allows us to decouple the sub-tasks of the planning domain by pruning the entire graph into disconnected components, each of which represents a sub-task. Two types of predicates are pruned from the causal graph: 1) predicates that have no object parameters are automatically removed, as they are always connected to all objects due to the symmetry of causal graph; 2) predicates that contains two or more objects as parameters will be pruned based on their values, as they represent coupling relationships between objects.

Fig. 3 shows the two types of pruned predicates in the object manipulation example: (free X) and any (unobstructed Y Z) evaluated to be true. By pruning the (free X) vertex, we relaxed the constraint that the robot arm can only be either empty or holding one object at the same time. For the purpose of task separation, this simplification does not lead to a significant loss of information because the (free X) constraint is still followed when each of the sub-tasks is solved. When (unobstructed A B) predicate is true, any robot manipulation of object A can be solved irrelevant to object B. Therefore, (unobstructed A B) can be pruned from the causal graph in order to explicitly decouple the predicates related to object A and B. In the example depicted in Fig. 3 (unobstructed A B) is false while all other unobstructed predicates are true.

The resulting pruned causal graph contains two disconnected components that contain sub-goals. This indicates that the full discrete planning can be divided into two sub-tasks.

After decomposition, the sub-tasks are serialized for a sequential solve. Here we assume no negative interactions between sub-tasks: 1) the actions in one sub-task do not delete sub-goals in another sub-task, and 2) the actions in one sub-task do not delete preconditions for actions in subsequent sub-tasks. Due to the pruning of causal graph, the second assumption might be violated if any pruned predicate is modified by the solution of a sub-task. Therefore, the causal graph is re-pruned after each sub-task is solved, and the remaining task is decomposed using the updated predicate values. If a new object coupling is introduced by a sub-plan, the additional related objects can be added to the respective remaining sub-tasks during re-pruning, which results in a final solution where an object might be manipulated multiple times in different sub-tasks.

In static environments, we assume that all robot motions are reversible. Therefore the robot motion in one sub-task does not cause a subsequent sub-task to become infeasible. In dynamic environments, the spontaneous changes of the environment means that committing to a sub-task might make a subsequent sub-task infeasible. For example, committing to grasping an object on the conveyor belt might make another object unreachable. To ensure completeness under dynamic environments, we use depth-first search to solve for sub-task sequence. However, the search can be significantly reduced by domain-specific heuristics to prioritize likely feasible sub-task sequences. In the conveyor belt domain, we prioritize the sub-task sequence as the order objects arrive.

**C. SYMBOLIC-DECISION-EMBEDDED BILEVEL OPTIMIZATION**

To solve for the lowest cost trajectory that achieves the symbolic sub-goal specified by PDDL and causal graph,
the TAMP problem is formulated as a bilevel optimization. Given initial and final symbolic states $s_0$, $s_K$ from the decomposed sub-task, the optimization will solve a sequence of discrete actions $A = \{a_1, \ldots, a_{K-1}\}$ resulting in a sequence of symbolic states $S = \{s_1, \ldots, s_K\}$, such that the total cost function $J$ is minimized. Meanwhile, between each symbolic state pair $(s_k, s_{k+1})$, a state trajectory $X_k = \{x_1, \ldots, x_{N_k}\}$ and a control trajectory segment $U_k = \{u_1, \ldots, u_{N_k-1}\}$ are optimized and the associated costs are incorporated into the high-level graph-search. $K$ denotes the number of discrete symbolic states and $N_k$ represents the number of knot points for the $k^{th}$ trajectory segment. This bilevel optimization is formulated as:

$$
\begin{align*}
\min_{s, A} & \sum_{k=0}^{K-1} \left[ J_{\text{path}}(s_k, a_k) + J_{\text{discr}}(s_k, a_k) \right] + J_{\text{goal}}(s_K) \\
\text{s.t.} & \quad s_0 = s_{\text{init}}, \ s_K = s_{\text{goal}}, \ a_k \in A(s_k), \\
& \quad h_{\text{switch}}(s_k, a_k) = 0, \ g_{\text{switch}}(s_k, a_k) \leq 0, \\
& \quad \min_{x_k, u_k} \sum_{i=0}^{N_k-1} L_{\text{path}}(x_i, u_i, a_k) \\
& \quad + L_{\text{goal}}(x_{N_k}, a_k) \\
& \quad s_{k+1} \in \mathcal{X}(a_k, u_k) \\
& \quad x_0 = s_{\text{init}}, \ x_{N_k} = s_{\text{goal}}, \ u_i \in \mathcal{U}_k, \forall i \in [0, N_k - 1] \\
& \quad \mathcal{X}(\cdot, \cdot) \ni \mathcal{U} \ni \mathcal{A} \\
& \quad g_{\text{obs}}(x_i) \geq 0 \\
& \quad \mathcal{X} \ni \mathcal{A} \ni \mathcal{U} \\
& \quad \forall i \in [0, N_k - 1]
\end{align*}
$$

where the path cost $J_{\text{path}}$ is composed of the local cost from lower-level trajectory optimization, i.e., $L_{\text{path}}$ and $L_{\text{goal}}$. $h_{\text{switch}}$ and $g_{\text{switch}}$ denote switch constraints given a symbolic state $s_k$ or an action $a_k$ that induces a state transition. $a_k \in A(s_k)$ indicates the set of all possible actions associated with a symbolic state $s_k$, $a_k$ is imposed over a specific trajectory segment in $\mathcal{X}$, which also governs the activeness of different sets of constraints corresponding to specific actions such as moving, pushing or throwing. Therefore, the symbolic-level transition is achieved through the continuous lower-level optimization $\mathcal{X}$.

The discrete cost $J_{\text{discr}}$ is defined at the discrete action level to encourage high-priority actions. In the conveyor belt example, if picking up one object has a higher priority over other objects, a higher discrete cost will be designed to penalize picking up other objects. To reflect the grasping priority, each object is assigned a base discrete cost $J_{\text{object}}$. At each symbolic node, the discrete costs for all unprocessed objects are scaled by a factor parameter $\alpha > 1$. Therefore, the sum of all object costs is $\sum_{p=0}^{P} \alpha^p J_{\text{object}, p}$, where $P$ is the number of objects. This cost is minimized to generate an optimal grasping sequence taking into account the object priority costs.

For the TO problem in $\mathcal{X}$, the associated action $a_k$ and current state $s_k$ determine the initial and goal states. The system dynamics constraint associated with the $k^{th}$ discrete mode is expressed as $f_k(x_i, u_i)$. The nonlinear function $g_{\text{obs}}(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^c$ is a minimum distance function that is decided by the collision checker to ensure collision avoidance, where $o$ is the number of collision pairs. We define feasible sets $\mathcal{X}_k$ and $\mathcal{U}_k$ to represent additional constraints such as the joint, torque and contact constraints subject to action $a_k$.

Note that the contact force is part of the state $x$ for contact modes with a grasped object. We solve the TO problem (1c) with DDP-ADMM. In particular, our DDP-ADMM consists of three blocks: (i) an unconstrained Differential Dynamic Programming (DDP) block, (ii) a constraint block handling constraints such as $X$ and $U$, and (iii) a collision avoidance block that solves a simple nonlinear program. Throughout the paper, DDP refers to the first stage in DDP-ADMM with only DDP block updated, while ADMM corresponds to the second stage that includes all the blocks. More details will be elaborated in Sec. [V].

D. SYMBOLIC MULTI-STAGE SEARCH

To solve the bilevel optimization problem defined in Sec. III-B, we extend greedy-best-first search to employ trajectory optimization as a subroutine to estimate path cost of each edge on the tree, where symbolic state $s_k$ activates and de-activates corresponding ADMM blocks to apply appropriate constraints. Each node $N$ of the search tree contains the information of its parent $P$, children $C_i \in C$, total cost-so-far, and the symbolic action leading up to the node, and the trajectory optimization solved on the edge between the node and its parent. For each node, we first use a geometric query module to compute the desired robot end-effector pose required by the action and check the kinematic feasibility of the action by inverse kinematics. If the action is feasible, we compute the trajectory cost from its parent using DDP and update its total path cost. Then a discrete exploration of search tree is done to find any symbolic goal node within a predefined number of discrete actions. Nodes to be explored are ranked in a priority queue $F$. When selecting a new node to visit, priority is given to the nodes with a symbolic goal node in its explored descendants, and with the lowest total cost from root. After a feasible solution is found by performing DDP only, the full ADMM is then used to refine the trajectories in order to comply with the kino-dynamic constraints. If a feasible trajectory cannot be found by full ADMM, the second best solution will be found via DDP and refined using ADMM again. This process is repeated until either an ADMM solution is generated, or the tree is fully explored. A pseudo code is depicted in Algorithm [I].

IV. DISTRIBUTED TRAJECTORY OPTIMIZATION

To enable discrete transitions in symbolic-level search for manipulation tasks, a distributed trajectory optimization – ADMM – is solved to generate dynamic-consistent motions at the low-level. The high-level symbolic actions will govern the activeness of different ADMM blocks in the low-level optimization where each ADMM block can be solved independently.
Algorithm 1 Multi-stage Search

**Input:** Root node $\mathcal{R}$

$\mathcal{F} \leftarrow \text{empty priority queue}$

$\mathcal{F}.\text{push}(\mathcal{R})$

while $\mathcal{F} \neq \emptyset$ do

$\mathcal{N} \leftarrow \mathcal{F}.\text{pop}()$

$\mathcal{P} \leftarrow \mathcal{N}.\text{parent}$

if $\text{InverseKinematics}(\mathcal{P}, \mathcal{N})$ is not feasible then

$\mathcal{J}_{\text{path}} \leftarrow \infty$

continue

end if

$\mathcal{J}_{\text{path}} \leftarrow \text{DDP}(\mathcal{P}, \mathcal{N})$

$\mathcal{N}.\text{cost} \leftarrow \mathcal{P}.\text{cost} + \mathcal{J}_{\text{path}} + \mathcal{J}_{\text{discrete}}$

if $\text{isGoal}(\mathcal{N})$ then

if ADMM($\mathcal{R}, \mathcal{N}$) is feasible then

return $\mathcal{N}$

end if

else

$\mathcal{C} \leftarrow \text{childNodes}(\mathcal{N})$

for $\mathcal{C}_i$ in $\mathcal{C}$ do

$\mathcal{C}_i.\text{hasGoalDescendent} \leftarrow \text{discreteExploration}(\mathcal{C}_i)$

$\mathcal{F}.\text{push}(\mathcal{C}_i)$

end for

end if

end while

A. OPERATOR SPLITTING VIA ADMM

We first review the basics of the ADMM approach. Consider a general two-block optimization problem with consensus constraints:

$$\min_{x, z} f(x) + g(z) \quad \text{s.t. } x = z$$

(2)

where two sets of variables $x$ and $z$ construct a separable cost function and a linear constraint. $f$ and $g$ can be non-smooth or encode admissible sets using indicator functions $[21]$. The ADMM algorithm splits the original problem into two blocks and iteratively updates the primal and dual variables as below until the convergence under mild conditions $[10]$.

$$x^{p+1} = \arg \min_x \left( f(x) + \frac{\rho}{2} \|x - z^p + w^p\|^2 \right)$$

(3a)

$$z^{p+1} = \arg \min_z \left( g(z) + \frac{\rho}{2} \|x^{p+1} - z + w^p\|^2 \right)$$

(3b)

$$w^{p+1} = w^p + x^{p+1} - z^{p+1}$$

(3c)

where $p$ denotes the ADMM iteration, $w$ is the scaled dual variable and $\rho$ is the penalty parameter.

Assuming that $g$ is an indicator function $I_B$ of a closed convex set $B$:

$$g(z) = I_B(z) = \begin{cases} 0, & z \in B \\ \infty, & \text{otherwise} \end{cases}$$

(4)

we can rewrite (3b) as

$$z^{p+1} = \arg \min_{z \in B} \left( \frac{\rho}{2} \|x^{p+1} - z + w^p\|^2 \right) = \Pi_B \left( x^{p+1} + w^p \right)$$

where $\Pi_B(\cdot)$ is a projection operator that projects the input argument onto an admissible set $B$ (see $[10]$ Sec. 4.1 and 5).

B. ADMM BLOCK DESIGN FOR MANIPULATION

To generate dynamically feasible trajectories given high-level manipulation actions, a set of ADMM blocks are constructed to match the manipulation predicates. In this section, we formulate the low-level ADMM-based trajectory optimizer for versatile manipulation.

As described in the previous subsection, the global optimization problem in Eq. (1) is composed of a high-level symbolic planner and a low-level trajectory optimizer. The low-level optimization problem is formulated as

$$\min_{X, U} \sum_{i=0}^{N} \mathcal{L}(x_i, u_i, a)$$

(6a)

subject to $x_0 = x_{\text{init}}, x_N = x_{\text{goal}}$, $x_{i+1} = f(x_i, u_i)$, $g_{\text{obs}}(x_i) \geq 0$, $X \in \mathcal{X}_a$, $U \in \mathcal{U}_a$, $a \in \mathcal{A}$

(6b)

where the action $a \in \mathcal{A}$ is sent from the high-level symbolic planner. Here we ignore the subscript $k$ for simplicity.

To save space, we use $\mathcal{L}$ to denote the low-level cost function comprising $\mathcal{L}_{\text{path}}$ and $\mathcal{L}_{\text{goal}}$ defined in (1). The design of the cost function $\mathcal{L}$ and additional constraints such as joint limits, torque limits and friction cone constraints, vary when different actions $a$ are active. The state is defined as $x = (q, \dot{q}, \lambda)^T$, where $q$ and $\dot{q}$ are the joint configuration and velocity vectors, respectively. When the manipulator performs nonprehensile actions such as pushing, the state $x$ includes the object states and $\lambda$ represents the stacked contact force. Otherwise, the object state and the contact force are dropped. The dimension of control input $u \in \mathbb{R}^m$ is always the same. The dynamics constraint $f$ represents rigid body dynamics and is numerically integrated by a 4th-order Runge-Kutta method. For a contact-involved action, a fully actuated manipulator with $n$ DoFs and a passive object with 6 DoFs will be modeled as the following

$$
\begin{bmatrix}
M(q) & 0_{6 \times 6} & 0_{6 \times n} \\
0_{n \times 6} & M_r(q_r) & 0_{n \times n} \\
0_{6 \times n} & 0_{n \times n} & I_{n \times n}
\end{bmatrix}
\dot{q} + \begin{bmatrix}
C(q) \frac{\partial C(q)}{\partial q} \\
0_{n \times 1}
\end{bmatrix} \tau + \begin{bmatrix}
M(q) \\
0_{n \times 1}
\end{bmatrix} u
$$

(7)

where the subscripts $o$ and $r$ represent the object and robot arm, respectively. $M \in \mathbb{R}^{(n+6) \times (n+6)}$ is the mass matrix; $C \in \mathbb{R}^{n+6}$ is the sum of centrifugal, gravitational, and Coriolis forces; $B \in \mathbb{R}^{(n+6) \times n}$ is the selection matrix for control inputs, which consists of a zero matrix for the object and an identity matrix for the manipulator; $F_{\text{ext}} \in \mathbb{R}^6$ denotes the external force applied on the object, such as the contact force exerted by the table in the pushing action. We define $\phi(q)$ as the signed distances between contact points...
and the object surface in the object’s frame. Then the stacked contact Jacobian matrix is expressed as \( J_{oc}(q) = \frac{\partial \phi}{\partial q} \).

Since the contact mode is known a priori in (6), a holonomic constraint on the acceleration with regard to the object frame can be established to compute the contact force along with the joint acceleration.

\[
J_{oc}\ddot{q} + J_{oc}\dot{q} = 0
\]

(8)

Given the rigid body dynamics in Eq. (7), the joint acceleration and contact forces are computed as:

\[
\ddot{q} = M^{-1}(-C + B\dot{q} + J_{oc}(q)^T\lambda)
\]

\[
\lambda = -(J_{oc}M^{-1}J_{oc}^T)^{-1}(J_{oc}\ddot{q} + \alpha J_{oc}\dot{q} + J_{oc}M^{-1}B\dot{q})
\]

where a restoring force \(-\alpha J_{oc}\dot{q}\) is added to mitigate the numerical constraint drifting in (6). The term \( J_{oc}M^{-1}J_{oc}^T \) is referred as the inverse inertia in contact space.

Given the manipulator dynamics and contact constraints above, the trajectory optimization in (6) can be further transformed for operator splitting. (Due to space limit, we ignore the summation notation and time-step subscripts in ADMM updates. For instance, \( \mathcal{L}(x, u, a) = \sum_{i=0}^{N} \mathcal{L}(x_i, u_i, a_i) \)).

\[
\min_{\phi, \phi, \phi} \mathcal{L}(x, u, a) + I_D(x, u) + I_G(\dot{x}, \ddot{u}) + I_{\Delta_x}\Delta_u(x, u)
\]

s.t. \( x = x, u = u, \dot{x} = \dot{x}, \ddot{u} = \ddot{u} \)

where \( D = \{(x, u) | x_0 = x_{init}, x_{i+1} = f(x_i, u_i), i = 0, 1, \ldots, N - 1 \} \) and \( G = \{(x, u) | g_{obs}(x_i) \geq 0, i = 0, 1, \ldots, N - 1 \} \) satisfy the dynamics constraint (6c) and collision avoidance constraint (6d), respectively. Note that the indicator function is generalized for non-convex sets similar to (11). For simplicity, \( \phi = (X, U) \) denotes the state-control pairs, while \( \phi = (\dot{X}, \dot{U}) \) contains the auxiliary variables to be projected onto admissible sets. The decision variable \( \phi = (X, U) \) is another set of auxiliary variables for handling collision avoidance. By making copies of the original state and control variables, the current optimization problem matches the general form in (2), where \( \phi, \phi \) correspond to \( x \), and \( \phi \) corresponds to \( z \).

The trajectories are updated in a distributed manner for the \( p^{th} \) ADMM iteration

\[
\phi^{p+1} = \arg \min_{\phi \in D} \mathcal{L}(x, u, a) + \frac{p_1}{2} \| x - \bar{x}^{p} + \bar{w}^{p} \|^2 \]

\[
+ \frac{p_u}{2} \| u - \bar{u}^{p} + \bar{w}^{p} \|^2
\]

(10a)

\[
\hat{\phi}^{p+1} = \arg \min_{\phi \in G} \frac{p_1}{2} \| \dot{x} - \bar{x}^{p} + \bar{w}^{p} \|^2 + \frac{p_u}{2} \| \dot{u} - \bar{u}^{p} + \bar{w}^{p} \|^2
\]

(10b)

\[
\bar{\phi}^{p+1} = \Pi_{\Delta_x}\Delta_u(x_i, u_i) (\frac{1}{2}(\phi^{p+1} + \bar{W} + \hat{\phi}^{p+1} + \bar{W}))
\]

(10c)

\[
W^{p+1} = \dot{W}_{p} + \phi^{p+1} - \hat{\phi}^{p+1}
\]

(10d)

\[
\dot{W}^{p+1} = \ddot{W}_{p} + \phi^{p+1} - \hat{\phi}^{p+1}
\]

(10e)

where (10a)-(10c) represent DDP, CollisionAvoidance (CA) and Projection sub-blocks. \( \dot{W}_{p} = (\dot{w}_j, w_u) \) and \( \dot{W}_{p} = (\dot{w}_j, \dot{w}_u) \) are stacked dual variables corresponding to DDP-Projection and CA-Projection consensus, respectively, with state constraints and torque limits dual variables. Since DDP solves unconstrained optimization efficiently, we use it to solve (10a). Due to the absence of \( \bar{u} \) in the collision avoidance constraint, (10b) reduces to:

\[
x^{p+1} = \arg \min_{g_{obs}(x) \geq 0} \frac{p}{2} \| x - \bar{x}^{p} + \bar{w}^{p} \|^2
\]

(11)

and we simply ignore the consensus of \( \bar{u} = \bar{u} \) in the ADMM updates as well. This collision-avoidance block is formulated as a nonlinear program and solved via IPOPT. For sub-problem (10c), a saturation function is employed to analytically project inputs onto admissible sets \( \chi_a \) and \( \mathcal{U}_a \), separately. Therefore, the optimization problem is decomposed into an unconstrained DDP block, a collision avoidance block and a projection block handling box constraints. Fig. 5 demonstrates the DDP-ADMM framework of our operator splitting method given the high-level action \( a_k \).

The DDP-ADMM solver has two stages corresponding to a multi-stage search scheme in Sec. III-D. The first stage only solves an unconstrained TO, i.e., the DDP block, for one time. The initial trajectory \( \phi^0 \) of this DDP solve is generated by a random guess for \( u \). As for Stage 2, a full ADMM is solved and the trajectory generated by Stage 1 is employed as a warm-start. Given this Stage 1 warm-start and the optimized \( \phi^0 \) from previous ADMM iteration, each DDP step in (10) only requires few iterations to converge (around 10 in most cases) within one ADMM iteration. The distributed structure of ADMM builds a natural connection between two optimization stages. The initial dual variables and trajectory for projection block \( \phi \) are arbitrarily selected. Here we initialize them with zeros. The ADMM stopping criterion is designed based on the residuals of different constraints

\[
\| x - \bar{x} \|^2 \leq \epsilon_x, \quad \| u - \bar{u} \|^2 \leq \epsilon_u, \quad \| \dot{x} - \ddot{x} \|^2 \leq \epsilon_{ca}
\]

where \( \epsilon_x, \epsilon_u, \) and \( \epsilon_{ca} \) are expected tolerances for state, control, and collision avoidance constraints.
actions, respectively. It is observed that in both cases, the
The two subfigures correspond to the along the entire trajectory with 100 knot points in Fig. 10.

optimizer, we show the normalized accumulated residuals
Box in sub-task 1 from either the side or the top.

costs bifurcate at action 12, where the robot grasps the red
actions. The selection of different action sequences results

tion, our planner finds 8 solutions for this task with distinct
hardware setup is shown in Fig. 7.

This task was evaluated
VOLUME 4, 2016
8

Therefore, the total planning time with the causal graph de-
two sub-tasks while eliminating the irrelevant black boxes
causal graph planner decomposes the planning problem into

If a black box obstructs the arm’s access to a target red box,

A. OBJECT SORTING IN CLUTTER
The goal of this object sorting task is to move all red boxes
from the cluttered area to the goal area as shown in Fig. 1(b).
If a black box obstructs the arm’s access to a target red box,
the black box needs to be moved away. In this case, the
causal graph planner decomposes the planning problem into
two sub-tasks while eliminating the irrelevant black boxes
from the sub-tasks. 6 objects out of 11 in total are grasped.
Therefore, the total planning time with the causal graph de-
composition is significantly reduced. This task was evaluated
in both the simulation and on the real robot hardware. The
hardware setup is shown in Fig. 7.

Using both sub-task sequence search and bilevel optimization,
our planner finds 8 solutions for this task with distinct
action sequences, each of which consists of 24 discrete
actions. The selection of different action sequences results
in different total costs, as shown in Fig. 8. For example, the
costs bifurcate at action 12, where the robot grasps the red
box in sub-task 1 from either the side or the top.

To evaluate the performance of our low-level trajectory
optimizer, we show the normalized accumulated residuals
along the entire trajectory with 100 knot points in Fig. 10.
The two subfigures correspond to the Move and Push
actions, respectively. It is observed that in both cases, the

accumulated residual for each constraint converges to high
accuracy, demonstrating satisfactory constraint violations.
The convergence performance varies in different constraint
set-ups, where the Push example has a tighter bound on
velocity limit. As a ballpark estimation, the DDP block, col-
sion avoidance block, and projection block approximately
take 81.1%, 18.5% and 0.066% of the total computation time,
respectively.

We evaluate the control performance of the object sorting
task both in Drake simulation and the hardware in Fig. 7.
Both the simulated and the real robot have a built-in PD
position controller with a servo rate of 1kHz. For a clear
visualization, we only show the trajectories of a short ma-
ipulation sequence of lifting an object, moving, and placing
it down (see Fig. 7). The trajectory depicts the Cartesian
end-effector position of the left fingertip. The desired and
the measured hardware trajectories have an average tracking
error of 1.9 cm throughout the pick-and-place motion.

B. CONVEYOR BELT SORTING
In our conveyor belt scenario, there are nine blocks cluttered
on a moving conveyor belt and four bins as shown in Fig. 1.
The blocks can take two different sizes (small, large), and
colors (red, black). The task is to sort the blocks from the
moving conveyor belt to the bins: small black blocks to Bin
2 and 3, and all red blocks to Bin 1 and 4. This problem
poses logical and dynamics constraints, because the large
blocks cannot be grasped, the small blocks can be grasped
in different poses, and Bin 4 is unreachable from the robot.
This leads to the necessity of pushing and throwing actions
shown in Fig. 6.
To evaluate the scalability of our planner, we compare the planning time with and without the causal graph decomposition for sorting objects on the conveyor belt with the first four objects in Fig. 1. Except for block pairs of D2-D3 and D4-D5, the objects are fully decoupled in the conveyor belt domain since they are not blocking each other. The results in Fig. 9 shows that the planning time grows linearly with the causal graph planner but exponentially with a single search tree. Note that the simulation result was generated by DDP without the ADMM refinement to compare only the computation time for expanding new tree nodes. In the causal graph decomposition, the size of search space depends on the coupling structure of the discrete predicates. In the extreme situation, all objects within the planning domain are decoupled, and then the total planning time with causal graph grows linearly with the number of objects. For real-world sequential manipulation, objects are often partially coupled. The causal graph decomposition will still offer computational advantages comparing to the conventional TAMP methods, depending on the level of multi-object coupling.

VI. DISCUSSION AND CONCLUSIONS
This study proposed a TAMP framework that optimized a sequence of kinodynamically consistent plans for a diverse set of manipulation tasks over a long-horizon. This framework is generalizable to other types of manipulation skills by rapidly adding specific constraints into trajectory optimization. One of our future directions will focus on maturing our trajectory optimization method: (i) employing advanced collision constraints to handle more complex geometries, and (ii) applying accelerated ADMM techniques to speed up the convergence [12].

One limitation of our current implementation stems from the heavy computational burden of trajectory optimization for highly complex manipulation motions, in particular, this optimization solve is coupled with the exploration of a large number of symbolic nodes during discrete search. To address this computation bottleneck, our future work will develop more computationally efficient TO algorithms through GPU-based parallel computing and Automatic Differentiation. As such, we can aim for online planning for reactive manipulation in dynamically changing environments.

The current framework requires a significant amount of human knowledge and effort. The initial values and transitions of predicates are user defined and the prunable predicates in causal graph needs to be explicitly defined for each domain. Additionally, manipulation parameters such as grasping locations are manually selected, resulting in a finite set of manipulation behaviors. This framework has the potential to be generalizable, by automating perception and state inference, learning related object importance for task decomposition [28] to automatically prune causal graph, learning compositional models for symbolic planning [29], and using optimization or sampling techniques to select manipulation parameters [30], with such as final object locations.

References


ZHIGEN ZHAO received the B.S. degree in mechanical engineering from Georgia Institute of Technology in 2018. He is currently pursuing a Ph.D. degree in Electrical and Computer Engineering at Georgia Institute of Technology. His research interest lies in the task and motion planning and machine learning algorithms applied in robot manipulation.

ZIYI ZHOU received the B.S. (2018) degree in automation from Northeastern University, Shenyang, China. He is currently pursuing a Ph.D. degree in Electrical and Computer Engineering at Georgia Institute of Technology. His research interest centers around contact-rich trajectory optimization and task planning applied on legged robots.

MICHAEL PARK received the B.S. degree in electrical and computer engineering from The University of Texas at Austin in 2016 and the M.S. degree in electrical and computer engineering from Georgia Institute of Technology in 2020 with a focus on machine learning. He is currently pursuing a Ph.D. in electrical and computer engineering at Georgia Institute of Technology with a focus on robotics. He has been a research engineer at Georgia Tech Research Institute since 2018, where he contributed to a number of projects as a task lead for robotic manipulation, machine learning, and computer vision.

YE ZHAO (S’10–M’19) received the B.E. degree in control science and engineering from Harbin Institute of Technology, Harbin, China, in 2011 and the M.S. and Ph.D. degrees in mechanical engineering from the University of Texas at Austin, Austin, TX, USA, in 2013 and 2016, respectively. He was a Postdoctoral Fellow with the John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA. He is currently an Assistant Professor at the George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology. He is affiliated to the Institute for Robotics and Intelligent Machines (IRIM). Dr. Zhao’s research interests include robust motion planning and contact-rich trajectory optimization, high-level task planning, decision-making algorithms, and legged robotic systems. He was a Co-Chair of the IEEE Robotics and Automation Society (RAS) Student Activities Committee and a Committee Member of IEEE-RAS Member Activities Board. He was also an ICREC Chair of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, and serves as an Associate Editor of IEEE Robotics and Automation Letters.