Fast MILP-based Task and Motion Planning for Pick-and-Place with Hard/Soft Constraints of Collision-Free Route

Takuma Kogo, Kei Takaya, and Hiroyuki Oyama

Abstract— We present new models of optimization-based task and motion planning (TAMP) for robotic pick-and-place (P&P), which plan action sequences and motion trajectory with low computational costs. We improved an existing state-of-the-art TAMP model integrated with the collision avoidance, which is formulated as a mixed-integer linear programing (MILP) problem. To enable the MILP solver to search for solutions efficiently, we introduced two approaches leveraging features of collision avoidance in robotic P&P. The first approach reduces number of binary variables, which are related to the collision avoidance of delivery objects, by reformulating them as continuous variables with additional hard constraints. These hard constraints maintain consistency by conditionally propagating binary values, which are related to the carry action state and collision avoidance of robots, to the reformulated continuous variables. The second approach is more aware of the branch-and-bound method, which is the fundamental algorithm of modern MILP solvers. This approach guides the MILP solver to find integer solutions with shallower branching by adding a soft constraint, which softly restricts a robot's routes around delivery objects. We demonstrate the effectiveness of the proposed approaches with a modern MILP solver.

I. INTRODUCTION

A. Background

Automation improves companies' productivity and competitiveness. Therefore, robots have been in increasing demand in industries, logistics, and other emerging domains for several years [1]. Currently, a major task of robots is handling (e.g., pick-and-place, packing, palletizing, etc.) [1]. In particular, pick-and-place (P&P), which is the sequential task of picking objects up from one location and placing them in a desired location, is a fundamental and essential task in many fields.

A basic approach to enable a robot to perform P&P is *teaching* where a human expert programs the robot's behavior with primitive commands on a step-by-step basis. *Teaching* is applicable to a variety of workspaces. However, slight changes to P&P specifications (e.g., layout of workspace, delivery object, etc.) require update works of *teaching* which is much time consuming. Therefore, *teaching-less* approaches have been desired to mitigate the aforementioned difficulties.

As a *teaching-less* approach, task and motion planning (TAMP) is a promising method [2]–[9]. TAMP can automatically generate action sequences and motion trajectory of robotic P&P with given information about the robot and workspace (Fig. 1). TAMP provides an advantage to robotic P&P consisting of discrete action sequences and continuous motion because it directly solves both of the plannings in

The authors are with Data Science Research Laboratories, NEC Corporation, Japan.
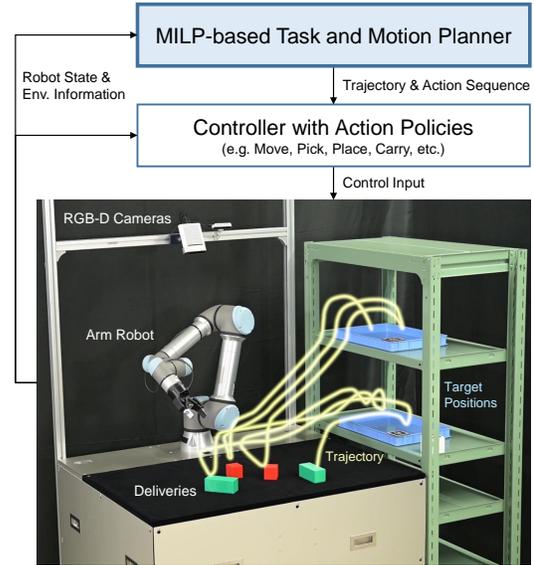
Fig. 1. Overview of system architecture. The arm robot delivers target objects (deliveries) to target positions, which are observed by the camera devices. The task and motion planner simultaneously outputs the motion trajectory and action sequence with the settings/observations of the robot and environment. The trajectory consists of discrete time-series data including the positions of the robot's end-effector (e.g., two finger gripper, vacuum hand, etc.). The action sequence consists of discrete time-series of symbolic actions (e.g. Move, Pick, Place, Carry, etc.) grounded in the mode transitions of system dynamics. The action policies are pre-designed for each symbolic action. The controller calculates the control input of the robot by interpolating the trajectory and converting the action sequence to action policies simultaneously.

a cooperative manner. Therefore, TAMP works especially for complex situations where non-trivial dependencies exist between the action sequences and motion trajectory.

Existing approaches for TAMP can be classified into two types: hierarchical feedback [2]–[5] and optimization-based [6]–[9]. In the hierarchical feedback approach [2]–[5], task planning and motion planning are alternately computed and provide feedbacks to each other to resolve infeasibility until a feasible motion plan is found. Although this approach has shown promising results, it requires parameter tuning on alternation control when applying it to various environments. In the optimization-based approach [6]–[9], TAMP is formulated as optimization models where a solution consists of both task and motion plans. This approach comparatively requires less parameter tuning for various environments because the optimization model is systematically designed and solved. However, the optimization-based TAMP requires a high computation cost for solving when a system's scale and/or complexity increases (e.g., collision avoidance).

B. Contributions

In this paper, we propose new models of optimization-based TAMP for robotic P&P, which plan action sequences and collision-free motion trajectory with low computation cost. Our motivation is reducing the computation cost of the optimization model based on the state-of-the-art TAMP [8] integrated with the state-of-the-art collision avoidance [11], which is formulated as a mixed-integer linear programming (MILP) problem. To the best of our knowledge, such an integrated model of MILP-based TAMP has not been fully studied in application to robotic P&P. Therefore, we propose two approaches leveraging features of collision avoidance in robotic P&P to reformulate the integrated model. Our contribution is presenting two models implemented with the approaches and their effectiveness as follows:

- The first proposed model reduces the number of binary variables related to the collision avoidance of deliveries. In this model, the binary variables are successfully reformulated as continuous variables with additional hard constraints. These hard constraints propagate the binary values, which are related to the carry action state and collision avoidance of robots, to the reformulated continuous variables.
- The second proposed model enables the MILP solver to find integer solutions more efficiently. In this model, a new term is added to the objective function as a soft constraint which softly restricts the robot's routes around a delivery object. This soft constraint guides the binary variables, which are related to collision avoidance with deliveries, to be fixed to the binary values in the MILP solver's search process.

II. Related Work

A. Task and Motion Planning

Optimization-based TAMPs have been proposed [6]–[9], and they are modeled as mathematical programming problems (i.e., mixed-integer nonlinear programming [6], [7], MILP [8], and continuous nonlinear programming [9]). The MILP-based TAMP proposed in [8] is formulated with constraints consisting of task conditions (e.g., completion, safety, etc.) and robot dynamics approximated with a low-dimensional mixed logical dynamical system. To achieve a plan in a low-dimensional space, pre-designed action policies are selected at a low-level control. We focus on this MILP-based TAMP as a promising method because optimization solvers have been developed to a practical level (See II-C). However, further study is needed for collision avoidance because it is not fully considered in the prior work.

B. Collision Avoidance

Much research has been conducted on collision avoidance. Many major approaches have been proposed and developed, such as potential field, random sampling, graph search, reachability analysis, optimization, etc. [10], [11], especially in the field of mobile robots. Collision avoidance problems are generally NP-hard, and practical methods basically rely
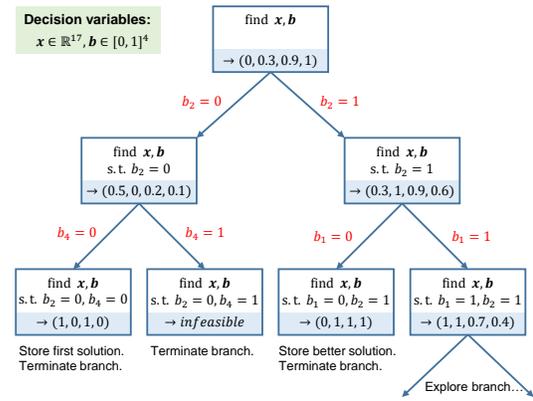

Fig. 2. Search example of B&B method. The basic strategy is a brute-force tree search recursively fixing binary variables one-by-one (*branching*) with early termination on the basis of theoretical conditions (*bounding*). In the B&B method, the continuous relaxation problem (CRP), where binary variables of the original MILP problem are converted to continuous 0-1 variables, is solved by linear programming. In a tree-search manner, the CRP is recursively updated with an additional constraint on a single original binary variable (highlighted in red) by using a fractional solution (shaded in blue) of the parent CRP. An incumbent solution is updated with a better integer solution until no more branches exist. *Branching* is terminated when any of the following conditions are satisfied: 1) an integer solution is found, 2) a CRP is infeasible, 3) a fractional solution is worse than the incumbent solution, and 4) a MIP gap, which is the difference between the objective values of the best and bound, is smaller than the setting value.

on heuristics for specific problems [10]. As for safety integrity, optimization-based collision-free methods have been proposed and applied [11]–[13]. These methods completely eliminate the cut-through issue caused by discrete-time models. In this paper, we focus on state-of-the-art collision-free encoding [11] (Fig. 4) as a promising method because it is compatible with MILP-based TAMP [8]. To the best of our knowledge, MILP-based TAMP integrated with collision-free formulation has not been fully studied, especially regarding computation efficiency. For all of these reasons, we address specific issues and leverage features of collision avoidance in the MILP-based TAMP of robotic P&P.

C. MILP Solver

MILP is becoming common to solve real-world problems owing to the following contributions: nonlinearity can be reasonably approximated to MILP by using well-known formulation techniques (e.g., big-M method), and MILP solvers have been over 3000 times algorithmically faster since 1988 [14]. Reports on modern commercial MILP solvers have also shown consistent speed-up and expansion of solvable problems since 2010 [15], [16]. However, MILP is NP-hard in general, and it requires a high computation cost when formulation is not appropriate for the branch-and-bound (B&B) method (Fig. 2). The B&B method is a fundamental algorithm for most modern MILP solvers [17]. Therefore, the following two aspects, which are empirically proven to impact search efficiency of B&B-based MILP solvers, are considered in the practice of formulation: *size* (e.g., number of binary variables) and *tightness* (e.g., feasible space of continuous relaxation problem) [17], [18]. We address both *size* and *tightness* with leveraging features of collision avoidance in the MILP-based TAMP of robotic P&P.
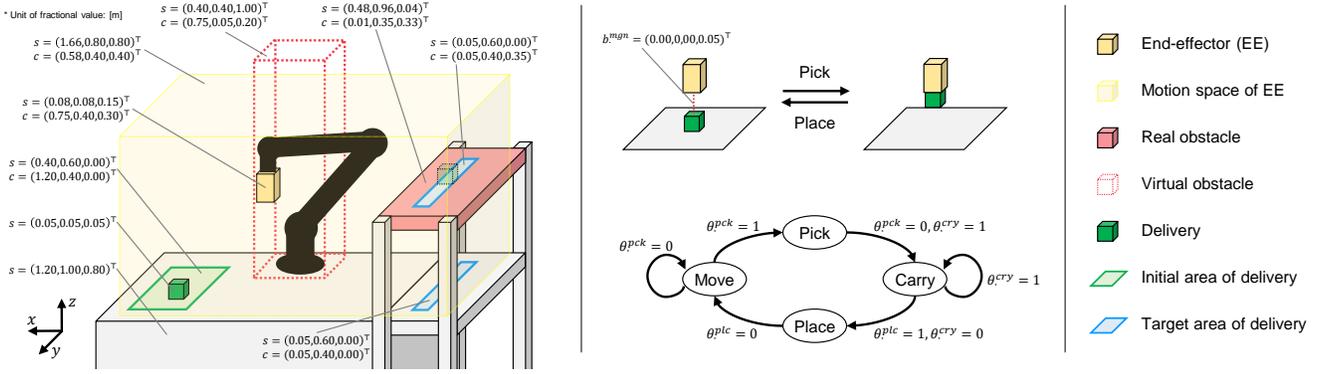
Fig. 3. Model of P&P planning. The right part shows the symbols of all modeled instances in this figure. The left part shows an example configuration of a P&P system with the instances and their parameters (shape width s and center position c). This configuration is also used in the evaluation. The following three simplifications are assumed to fit the computation cost within a practical range: 1) the instances are modeled as axis-aligned bounding boxes (AABB), 2) the dynamics of a robot's body part (black object) is not modeled, 3) irrelevant obstacles (gray objects) outside the motion space of the end-effector are not modeled. The center part shows state transitions of the delivery (top) and the end-effector (bottom) regarding the actions. The end-effector and grasping delivery face each other on the bottom and top surfaces, respectively. Moreover, the end-effector takes margin $b_\bullet^{mgn}$ between itself and the grasping delivery at the pick/place actions.

III. BASELINE MODEL

A. System Architecture

Fig. 1 shows an overview of our system architecture. Our system architecture is based on prior studies [8], [9] and it has the only difference in regard to the model of the task and motion planner. In this paper, we assumed a single arm robot configuration for a simple explanation and evaluation. Note, however, that multi-arm robot configurations are also supported by the model of the task and motion planner as formulated in the following subsections III-C and III-D.

B. Modeling Settings

In this paper, we introduce our baseline model of the task and motion planner for a robotic P&P. Our baseline model is based on the MILP-based TAMP of [8]. Moreover, our baseline model is newly integrated with collision-free encoding [11] for safer collision avoidance. In addition, our baseline model has the following minor improvements:

- Simple formulation of delivery's dynamics with an approach point (Eqs. (7)–(13), center of Fig. 3).
- Strengthened formulation for search efficiency of MILP solver (Eqs. (16)–(17)).
- Direct minimization of time steps to complete P&P with consideration of distance minimization (Eqs. (26)–(30)).

Fig. 3 illustrates our baseline model formulated as a MILP problem. The purpose of the baseline model is to generate motion trajectory and action sequences of the end-effector by solving it. The concrete formulation with the constraints and objective function is explained in the following subsections.

C. Notation

$N_{stp}, N_{ee}, N_{dlv}, N_{obs}$, and N_{rg}, where $N_\bullet \in \mathbb{N}_+$ are constants, denote the number of time steps, end-effectors, deliveries, obstacles, and collision-free regions, respectively. Let $\bullet$ denote any string. t, i, $j_\bullet$, k, and r respectively represent the following indices: "t-th time step", "i-th end-effector", "$j_\bullet$-th delivery", "k-th obstacle", and "r-th collision-free region". Moreover, we define their indices as $t \in \{0, ..., N_{stp}\}$,

$i \in \{1, ..., N_{ee}\}$, $j_\bullet \in \{1, ..., N_{dlv}\}$, $k \in \{1, ..., N_{obs}\}$, and $r \in \{1, ..., N_{rg}\}$ where $j_1 \neq j_2$. We give all possible combinations of the indices for each constraint and objective function. The upper/lower bounds of $\bullet$ are constants denoted by $\bar{\bullet}$ and $\underline{\bullet}$, respectively. The initial/target values of $\bullet_t$ are also constants denoted by $\bullet_0$ and $\bullet_\star$, respectively. Furthermore, the relative value of $\bullet_t$ from its initial value is denoted by $\bullet_t'$ (i.e., $p_t' = p_t - p_0$). For simple formulation, we use the logical operators "NOT" $\neg$, "AND" $\wedge$, and "OR" $\vee$, which are transformed to the equivalent MILP formulation with additional decision variables as shown in Eqs. (1)–(3):

$$\phi = \neg\varphi \quad \Leftrightarrow \quad \phi = 1 - \varphi, \tag{1}$$

$$\phi = \bigwedge_n \varphi_n \Leftrightarrow \begin{cases} \phi \leq \varphi_n \\ \phi \geq \sum_n \varphi_n - N + 1 \end{cases} n \in \{1, ..., N\}, \tag{2}$$

$$\phi = \bigvee_n \varphi_n \Leftrightarrow \begin{cases} \phi \geq \varphi_n \\ \phi \leq \sum_n \varphi_n \end{cases} n \in \{1, ..., N\}, \tag{3}$$

where $\phi \in [0, 1]$ is the additional decision variable and $\varphi_n \in \{0, 1\}$ is the operand ($\varphi_n \in [0, 1]$ is allowable).

D. Formulation

Equations (4)–(29) are the constraints of the baseline model. Equation (30) is the objective function of the baseline model. The baseline model is a discrete bounded time system where the time step size Δt and its number N_{stp} are respectively given as constant values.

Constraint (4) represents the dynamics of the end-effector:

$$p_{i,t+1}^{ee} = p_{i,t}^{ee} + v_{i,t}^{ee}\Delta t \tag{4}$$

where $p_{i,t}^{ee}, v_{i,t}^{ee} \in \mathbb{R}^3$ are the position and velocity of the end-effector, respectively. For simplification, we use such a discrete state-space model where the control input is velocity as the dynamics of the end-effector.

Constraints (5) and (6) represent the feasible interaction among the end-effectors and deliveries, which means "one

end-effector can grasp one delivery at the same time":

$$\sum_i \theta_{i,j,t}^{gsp} \leq 1, \tag{5}$$

$$\sum_j \theta_{i,j,t}^{gsp} \leq 1 \tag{6}$$

where $\theta_{i,j,t}^{gsp} \in \{0,1\}$ is the action state where the end-effector is grasping the delivery.

Constraints (7)–(9) represent the states corresponding to the action sequences of the end-effector:

$$\theta_{i,j,t}^{pck} = (\neg\theta_{i,j,t}^{gsp}) \wedge \theta_{i,j,t+1}^{gsp}, \tag{7}$$

$$\theta_{i,j,t}^{plc} = (\neg\theta_{i,j,t}^{gsp}) \wedge \theta_{i,j,t-1}^{gsp}, \tag{8}$$

$$\theta_{i,j,t}^{cry} = \theta_{i,j,t}^{gsp} - \theta_{i,j,t}^{pck} \tag{9}$$

where $\theta_{i,j,t}^{pck}, \theta_{i,j,t}^{plc}, \theta_{i,j,t}^{cry} \in [0,1]$ are the action states where the end-effector is picking, placing, and carrying the delivery, respectively (See Fig. 3). These constraints decompose and propagate the binary values of $\theta_{i,j,t}^{gsp}$ to $\theta_{i,j,t}^{pck}$, $\theta_{i,j,t}^{plc}$, and $\theta_{i,j,t}^{cry}$.

Constraints (10)–(13) represent the dynamics of the delivery as shown in Fig. 3:

$$-M_1(\neg\theta_{i,j,t}^{cry}) \leq p_{j,t}^{dlv} - p_{i,t}^{ee} + b_{i,j}^{on} \leq M_1(\neg\theta_{i,j,t}^{cry}), \tag{10}$$

$$-M_2(\neg\theta_{i,j,t}^{pck}) \leq p_{j,t}^{dlv} - p_{i,t}^{ee} + b_{i,j}^{off} \leq M_2(\neg\theta_{i,j,t}^{pck}), \tag{11}$$

$$-M_3(\neg\theta_{i,j,t}^{plc}) \leq p_{j,t}^{dlv} - p_{i,t}^{ee} + b_{i,j}^{off} \leq M_3(\neg\theta_{i,j,t}^{plc}), \tag{12}$$

$$-M_4\sum_i \theta_{i,j,t}^{gsp} \leq p_{j,t+1}^{dlv} - p_{j,t}^{dlv} \leq M_4\sum_i \theta_{i,j,t}^{gsp} \tag{13}$$

where $p_{j,t}^{dlv} \in \mathbb{R}^3$ is the position of the delivery. Moreover, $b_{i,j}^{on}, b_{i,j}^{off} \in \mathbb{R}^3$ are constants representing the relative position between the delivery and the end-effector when the delivery is carried and not carried, respectively. The concrete values of the constants are given by $b_{i,j}^{on} = (0, 0, \frac{1}{2})^\top \circ (s_i^{ee} + s_j^{dlv})$, $b_{i,j}^{off} = b_{i,j}^{on} + b_{i,j}^{mgn}$ where $\circ$ is the operator of an element-wise product and $s_i^{ee}, s_j^{dlv} \in \mathbb{R}_+^3$ are constants representing the shape width of the end-effector and delivery, respectively. $M_1, ..., M_4$ are large coefficients to switch dynamics via inequality constraints with binary variables, and we give the tightest value to each coefficient of the constraints.

Constraints (14) and (15) represent the P&P completion conditions for each of the deliveries:

$$-M_5(\neg\psi_{j,t}) \leq p_{j,t}^{dlv} - p_{j,\star}^{dlv} \leq M_5(\neg\psi_{j,t}), \tag{14}$$

$$\psi_{j,t} \leq \neg\sum_i \theta_{i,j,t}^{gsp} \tag{15}$$

where $\psi_{j,t} \in \{0,1\}$ is the state where the delivery is at its target position without being grasped, and M_5 is a large coefficient similar to $M_1, ..., M_4$.

Constraints (16) and (17) force $\psi_{j,t}$ to take 1 when the delivery is at its target position by preventing $\psi_{j,t}$ from taking 0. As a result, the computation cost of the MILP solver can be substantially reduced. Constraint (16) means that the delivery must be at its target position when grasping is finished. Constraint (17) means that the delivery keeps its position once it reached its target position.
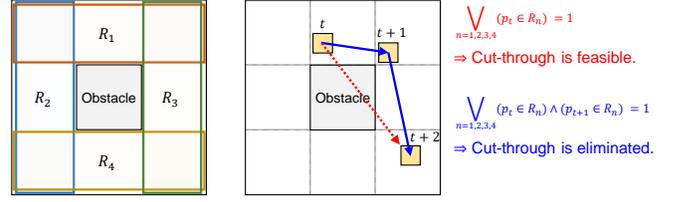

Fig. 4. Model of collision avoidance [11]. The left part shows a 2D space example of a collision-free region, which is defined as AABB on each side of the obstacle (i.e., $R_1, ..., R_4$). The important point is that each collision-free region overlaps with adjacent regions (e.g., R_2 overlaps with R_1 and R_4). The right part shows a collision-free path and constraint (highlighted in blue) compared with the case of cutting through the obstacle (highlighted in red). The bottom constraint requires that the moving object is positioned within any one of the same collision-free regions at consecutive time steps. This constraint completely eliminates the cut-through of corners.

$$\psi_{j,t+1} - \psi_{j,t} \geq \sum_i (\theta_{i,j,t}^{gsp} - \theta_{i,j,t+1}^{gsp}), \tag{16}$$

$$\psi_{j,0} \leq \psi_{j,1} \leq ... \leq \psi_{j,N_{stp}} = 1. \tag{17}$$

Constraints (18)–(29) represent the collision-free encoding [11] as shown in Fig. 4. The collision avoidance of the end-effector with the obstacle is modeled by constraints (18) and (22), and that with the delivery is modeled by constraints (19) and (23). The collision avoidance of the delivery with the obstacle is modeled by constraints (20) and (24), and that with the other delivery is modeled by constraints (21) and (25). On the basis of the big-M method, constraints (18)–(21) evaluate whether a moving object (i.e., end-effector and delivery) is within the collision-free region:

$$p_{i,t}^{ee} \leq \overline{R_{k,r}^{obs}} z_{i,k,r,t}^{ee,obs} + \overline{p_{i,t}^{ee}}(\neg z_{i,k,r,t}^{ee,obs}),$$
$$p_{i,t}^{ee} \geq \underline{R_{k,r}^{obs}} z_{i,k,r,t}^{ee,obs} + \underline{p_{i,t}^{ee}}(\neg z_{i,k,r,t}^{ee,obs}) \tag{18}$$

where $R_{k,r}^{obs} \in \mathbb{R}^3$ is the position of the collision-free region on the side of the obstacle and $z_{i,k,r,t}^{ee,obs} \in \{0,1\}$ is the state where the end-effector is within the corresponding collision-free region on the side of the obstacle.

$$p_{i,t}^{ee} - p_{j,t}^{dlv'} \leq \overline{R_{j,r}^{dlv}} z_{i,j,r,t}^{ee,dlv} + (\overline{p_{i,t}^{ee} - p_{j,t}^{dlv'}})(\neg z_{i,j,r,t}^{ee,dlv}),$$
$$p_{i,t}^{ee} - p_{j,t}^{dlv'} \geq \underline{R_{j,r}^{dlv}} z_{i,j,r,t}^{ee,dlv} + (\underline{p_{i,t}^{ee} - p_{j,t}^{dlv'}})(\neg z_{i,j,r,t}^{ee,dlv}) \tag{19}$$

where $R_{j,r}^{dlv} \in \mathbb{R}^3$ is the position of the collision-free region on the side of the delivery and $z_{i,j,r,t}^{ee,dlv} \in \{0,1\}$ is the state where the end-effector is within the corresponding collision-free region on the side of the delivery.

$$p_{j,t}^{dlv} \leq \overline{R_{k,r}^{obs}} z_{j,k,r,t}^{dlv,obs} + \overline{p_{j,t}^{dlv}}(\neg z_{j,k,r,t}^{dlv,obs}),$$
$$p_{j,t}^{dlv} \geq \underline{R_{k,r}^{obs}} z_{j,k,r,t}^{dlv,obs} + \underline{p_{j,t}^{dlv}}(\neg z_{j,k,r,t}^{dlv,obs}) \tag{20}$$

where $z_{j,k,r,t}^{dlv,obs} \in \{0,1\}$ is the state where the delivery is within the collision-free region on the side of the obstacle.

$$p_{j_1,t}^{dlv} - p_{j_2,t}^{dlv'} \leq \overline{R_{j_2,r}^{dlv}} z_{j_1,j_2,r,t}^{dlv,dlv} + (\overline{p_{j_1,t}^{dlv} - p_{j_2,t}^{dlv'}})(\neg z_{j_1,j_2,r,t}^{dlv,dlv}),$$
$$p_{j_1,t}^{dlv} - p_{j_2,t}^{dlv'} \geq \underline{R_{j_2,r}^{dlv}} z_{j_1,j_2,r,t}^{dlv,dlv} + (\underline{p_{j_1,t}^{dlv} - p_{j_2,t}^{dlv'}})(\neg z_{j_1,j_2,r,t}^{dlv,dlv}). \tag{21}$$

where $z_{j_1,j_2,r,t}^{dlv,dlv} \in \{0,1\}$ is the state where the delivery is within the collision-free region on the side of the other obstacle.

Constraints (22)–(25) force the moving objects to be within any one of the same collision-free regions at two consecutive time steps to completely prevent cut-through:

$$\bigvee_r (z_{i,k,r,t}^{ee,obs} \wedge z_{i,k,r,t+1}^{ee,obs}) = 1, \tag{22}$$

$$\bigvee_r (z_{i,j,r,t}^{ee,dlv} \wedge z_{i,j,r,t+1}^{ee,dlv}) = 1, \tag{23}$$

$$\bigvee_r (z_{j,k,r,t}^{dlv,obs} \wedge z_{j,k,r,t+1}^{dlv,obs}) = 1, \tag{24}$$

$$\bigvee_r (z_{j_1,j_2,r,t}^{dlv,dlv} \wedge z_{j_1,j_2,r,t+1}^{dlv,dlv}) = 1. \tag{25}$$

Constraints (26) and (27) are for exploiting the completion time steps of all P&P and moving distances of the end-effectors to be used in the objective function:

$$C_t = \bigwedge_{\tau=t,\dots,N_{stp}} \bigwedge_j \psi_{j,\tau}, \tag{26}$$

$$-u_{i,t} \le v_{i,t}^{ee} \le u_{i,t} \tag{27}$$

where $C_t \in [0,1]$ is the state where the P&P of all deliveries is completed and $u_{i,t} \in \mathbb{R}_{\ge 0}^3$ is a slack variable to obtain the absolute value of $v_{i,t}^{ee}$ by combining it with the objective function (i.e., minimizing $u_{i,t}$).

Constraints (28) and (29) represent the terms of the objective function.

$$J_{time} = \frac{1}{N_{stp}+1} \sum_t (\neg C_t), \tag{28}$$

$$J_{dist} = \sum_t \sum_i w_t \|u_{i,t}\|_1 \tag{29}$$

where J_{time} is the normalized time steps to complete all P&P and J_{dist} is the weighted sum of the L1 distance where the end-effectors move. $w_t \in \mathbb{R}_+$ is a small weight coefficient to prioritize J_{time} over J_{dist} to evaluate minimality. We give conservative values to w_t by $w_t = (1+\alpha)^{\frac{t}{N_{stp}}-1} / ((N_{stp}+1)^2 \sum_i \overline{\|v_{i,t}^{ee}\|_1})$ where $\alpha \in \mathbb{R}_+$ is a penalty coefficient. These w_t guarantee $\frac{1}{N_{stp}+1} \ge J_{dist}$ where the minimum variation of J_{time} is greater than the maximum variation of J_{dist}. In addition, these w_t contribute to generate reasonable motion trajectory because moves are penalized at later time steps.

Equation (30) represents the objective function of the baseline model:

$$X^* = \arg\min_X (J_{time} + J_{dist}) \tag{30}$$
$$\text{s.t.} \quad Eq.\ (4)\text{–}(29)$$

where X^* is the optimal solution of X which is a vector consisting of all decision variables (e.g., $p_{i,t}^{ee}$, $v_{i,t}^{ee}$, $\theta_{i,j,t}^{pck}$, $\theta_{i,j,t}^{plc}$, $\theta_{i,j,t}^{cry}$, etc.).

The model can be solved by the MILP solver. Therefore, the optimal $p_{i,t}^{ee}$, $v_{i,t}^{ee}$, $\theta_{i,j,t}^{pck}$, $\theta_{i,j,t}^{plc}$ and $\theta_{i,j,t}^{cry}$ can be extracted from X^* as the trajectory and actions sequences.

TABLE I
COMPARISON OF MODELS

Model	Variable $z_\bullet^{dlv,\bullet} \in \dots$	Constraints Eq. (4)–(23),(26)–(29),...	Objective value $J_{time}+J_{dist}\cdots$
Baseline	$\{0,1\}$	Eq. (24), (25)	
Ours (hard)	$[0,1]$	Eq. (31), (32)	
Ours (hard+soft)	$[0,1]$	Eq. (31)–(33)	$+J_{route}$

IV. PROPOSED APPROACH

A. Motivation and Overview

Our motivation is to reduce the computation cost of the baseline model via leveraging features of collision avoidance in robotic P&P. To the best of our knowledge, few prior studies have addressed such work. We take two approaches improving the *size* and *tightness* of the baseline model, which impact the search efficiency of the MILP solver as mentioned in subsection II-C. The purpose of the first approach regarding *size* is to reduce the search space of the MILP solver. The purpose of the second approach regarding *tightness* is to make *bounding* work more efficiently, in other words, to guide the MILP solver to find integer solutions at shallower nodes in the search tree. The policy of the first approach is to reduce the number of binary variables in regard to collision avoidance of the deliveries. The policy of the second approach is to softly tighten the values of binary variables in regard to collision avoidance with the deliveries. In this paper, we implement these approaches by using continuous relaxation with hard constraints and a soft constraint, respectively. Table I shows a detailed comparison of the baseline and proposed models. "Ours (hard)" is based on the first approach, and "Ours (hard+soft)" is based on both of the first and second approaches.

B. Hard Constraint

For the first approach, we introduced an assumption where the collision-free regions of the end-effector and carried delivery are identified. This assumption can be simply formulated as the following additional constraints:

$$z_{j,k,r,t}^{dlv,obs} = \bigvee_i (z_{i,k,r,t}^{ee,obs} \wedge \theta_{i,j,t}^{cry}), \tag{31}$$

$$z_{j_1,j_2,r,t}^{dlv,dlv} = \bigvee_i (z_{i,j_2,r,t}^{ee,dlv} \wedge \theta_{i,j_1,t}^{cry}). \tag{32}$$

As a result, the values of $z_{j,k,r,t}^{dlv,obs}$ and $z_{j_1,j_2,r,t}^{dlv,dlv}$ are conditionally bound with values of $z_{i,k,r,t}^{ee,obs}$ and $z_{i,j_2,r,t}^{ee,dlv}$, respectively. In other words, $z_{j,k,r,t}^{dlv,obs}$ and $z_{j_1,j_2,r,t}^{dlv,dlv}$ can be reformulated as 0-1 continuous variables. For all r, their values take 0 when the delivery is carried by no end-effector. This possibly induces an infeasibility of the collision-free encoding, but it can be resolved by removing constraints (24) and (25) from the set of constraints. In this case, the lack of constraints (24) and (25) does not affect the collision avoidance of the delivery because the delivery always stays within any collision-free regions when it is not carried by the end-effectors. This insight also plays an important role of the first approach. Table I shows a summary of the aforementioned description as "Ours (hard)".
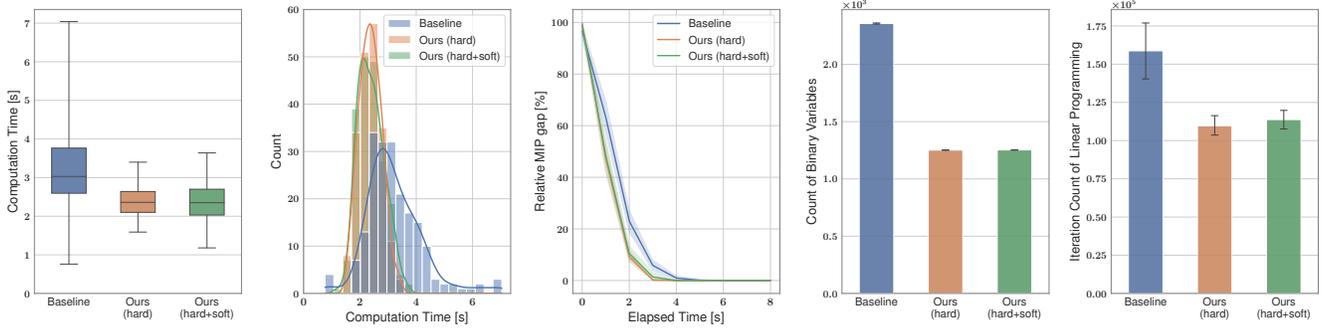
Fig. 5. Result of optimization in the case of $N_{dlv} = 2$ ($N_{stp} = 30$). The graphic representation is as follows: 1) boxplot, 2) histogram (translucent bar) with kernel density estimation (solid line), 3) time series of the mean (solid line) with 99% confidence intervals (shaded area), and 4), 5) bar chart of the mean (solid bar) with 99% confidence intervals (black bold bar), respectively. Computation time means the time to find an optimal solution. Binary variables were counted after the models were presolved. Iteration count of linear programming means count of CRPs solved by the MILP solver.
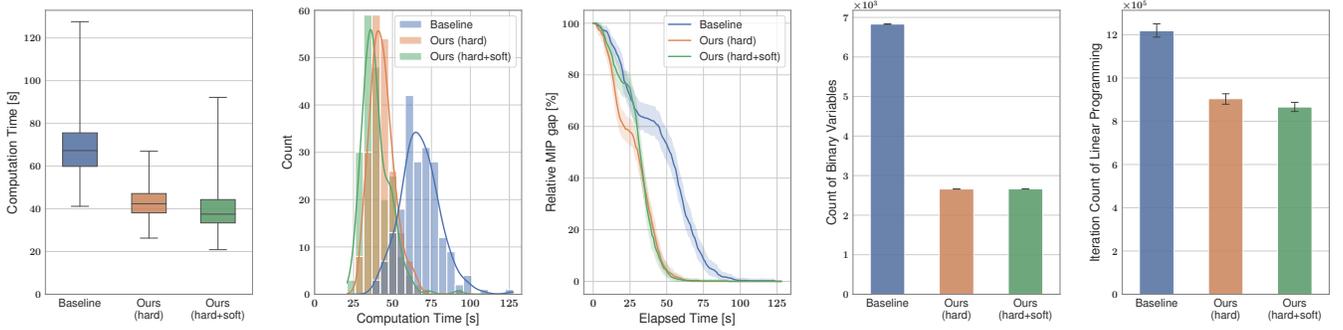

Fig. 6. Result of optimization in the case of $N_{dlv} = 3$ ($N_{stp} = 45$). The graphic representation is in a similar manner with Fig. 5.

C. Soft Constraint

For the second approach, we introduced another assumption where it is feasible to softly restrict the end-effector's routes around the deliveries. The reason we choose such a soft constraint is that it is difficult to preliminarily determine the collision-free regions not to be passed by the end-effector as a hard constraint. The second assumption can be also simply formulated as follows:

$$J_{route} = \sum_t \sum_i \sum_j \sum_{r \in R_{i,j,t}^c} z_{i,j,r,t}^{ee,dlv}, \quad (33)$$

$$X^* = \arg\min_X \left(J_{time} + J_{dist} + J_{route} \right) \quad (34)$$
$$\text{s.t.} \quad Eq.\ (4)\text{--}(23), (26)\text{--}(29), (31)\text{--}(33)$$

where J_{route} is the sum of the penalties for the end-effector to pass through the collision-free regions on the sides of delivery, which are selected from $R_{i,j,t}^c$. Moreover, $R_{i,j,t}^c$ is the set of the collision-free regions around delivery, which is pre-configured on the basis of the predicted probability of the pass-through. In this paper, we simply configured all $R_{i,j,t}^c$ to contain three collision-free regions as follows: the first is on the bottom side of the delivery and the remaining are on both sides of the delivery along the horizontal axis where the end-effector moves less (i.e., y-axis in Fig. 3). This simple way is based on the prior knowledge about routes in robotic P&P where the end-effector typically moves over the deliveries and along a long axis direction. Note that a new term J_{route} is added to the objective function. J_{route} is minimized most preferentially because the minimum variation of J_{route} is greater than the maximum variation of $J_{time} + J_{dist}$. As a result, this soft constraint makes the MILP solver find fractional solutions consisting of more $z_{i,j,r,t}^{ee,dlv} = 0$. In other words, the MILP solver has more chances of *bounding* where the integer solutions are found. The constraint is inherited from "Our (hard)" for the evaluation. Table I shows a summary of the aforementioned description as "Ours (hard+soft)".

V. EVALUATION

A. Settings

We compare the baseline model and proposed models as shown in Table I. All of the models use the same configuration with the parameters shown in Fig. 3 and the remaining parameters are as follows: $\Delta t = 0.50$ s, $N_{stp} = 15 N_{dlv}$, $N_{dlv} \in \{2, 3\}$, $\overline{|v_{i,t}^{ee}|} = (0.40, 0.20, 0.20)^\top$ m/s, and $\alpha = 1$. We randomly sampled 200 combinations of initial and target positions of the deliveries, which are within the area shown in Fig. 3. We solved all of the models implemented by Python-mip 1.13.0 [19] with the MILP solver Gurobi 9.1.1 [20] on an Intel Core i7-10700K (8-core/3.80GHz) and 16-GB RAM. We basically used the default settings of Gurobi in 8-thread parallel execution and 300 s time limit, but disabled cut generation and feasibility pumping, which equally slowed computation down for all of the models.

B. Results

Fig. 5 shows the result in the case of $N_{dlv} = 2$ ($N_{stp} = 30$). The boxplot shows that both of the proposed models reduced the computation time by about 23% in the range of the 25–75th percentiles. Similarly, the histogram shows that both the mean and variance of the computation time are improved. However, there is only a slight difference of the variance compared with that of the mean between the proposed models. The time-series data of the relative MIP gap shows that the proposed models have better convergence properties. The left bar chart shows that the number of binary variables were reduced by about 50% in the proposed models. The right bar chart shows that the iteration count of linear programming was reduced by about 31% in the proposed models. As a result, the computation time was reduced by about 23%. We can summarize that our first approach, which reduces the number of binary variables, certainly improved computation time but our second approach did not contribute in the case of $N_{dlv} = 2$.

Fig. 6 shows the result in the case of $N_{dlv} = 3$ ($N_{stp} = 45$). These graphs show similar findings to those of $N_{dlv} = 2$ (Fig. 5) and much greater improvement over the baseline model. We examined the cause of the improvement, where the reduction rate of the computation time (about 38, 46%) was greater than the reduction rate of the iteration count (about 25, 30%). We found that the number of constraints were also considerably reduced in the proposed models because of the removal of constraints (24) and (25). It is supposed that small *size* model resulted in small computation time per iteration and it contributed to much further speed-up. In addition to that, the computation time was improved even more by the second approach. The right bar chart shows greater improvement in search efficiency by "Ours (hard+soft)" and this was not observed in the case of $N_{dlv} = 2$. It is supposed that this cumulative speed-up can be larger when N_{dlv} is larger. According to this result and the boxplot, the proposed models independently address the computation efficiency of collision avoidance with deliveries. We can summarize that our second approach, which softly tightens binary variables, further reduces computation time when N_{dlv} is larger.

Fig. 7 and 8 show the results of the sensitivity analysis on parameter N_{stp} in the case of $N_{dlv} = 2, 3$, respectively. We varied N_{stp} from a tight setting (J_{time} close to 1) to a loose setting (smaller J_{time}) for the use case where N_{stp} cannot always be configured with a tight setting. The left graph shows that the proposed models robustly outperform the baseline model in the range. Furthermore, in the case of $N_{dlv} = 3$, "Ours (hard+soft)" is faster than "Ours (hard)" marginally but robustly. The right graph shows that only "Ours (hard+soft)" resulted in slight degradation regarding the completion time steps J_{time}. This is the disadvantage of the second approach but its impact is acceptably small in practice. We successfully validated effectiveness of the proposed models in the evaluation configuration which is a typical P&P scenario. However, further investigation with
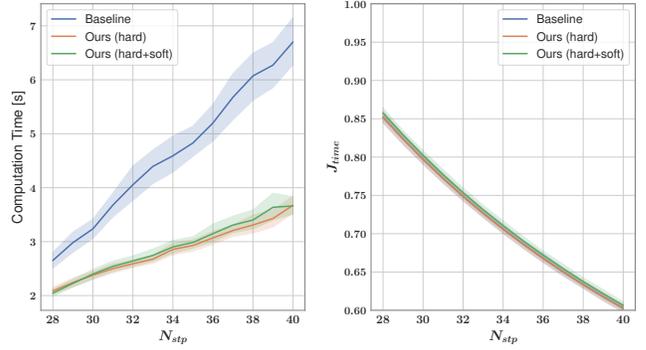

Fig. 7. Sensitivity analysis on parameter N_{stp} in the case of $N_{dlv} = 2$. Each solid line represents the mean value. Each shaded area shows 99% confidence intervals. Computation time means time to find an optimal solution.
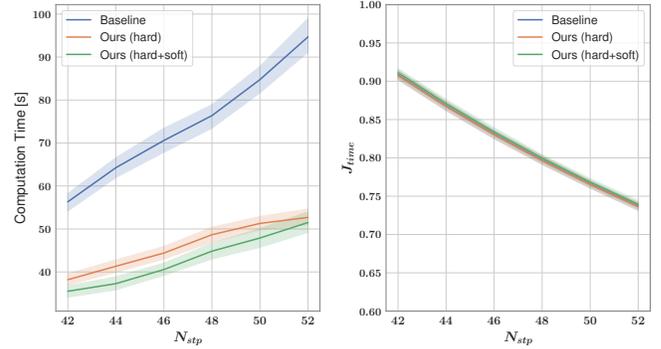

Fig. 8. Sensitivity analysis on parameter N_{stp} in the case of $N_{dlv} = 3$. The graphic representation is in a similar manner with Fig. 7.

more complex configurations (i.e., layout of obstacle, delivery, multi-arm robot, etc.) is needed. This is a future work to clarify how much the proposed model relies on a tradeoff between the completion time steps J_{time} and the computation time.

VI. CONCLUSION

We presented new models of MILP-based TAMP for robotic P&P, which plans action sequences and motion trajectory with low computation costs. We improved an existing state-of-the-art MILP-based TAMP model integrated with collision avoidance. To enable the MILP solver to search for solutions efficiently, we introduced two approaches leveraging features of collision avoidance in robotic P&P. The first approach reduces the number of binary variables, which are related to the collision avoidance of delivery objects, by reformulating them as continuous variables with additional hard constraints. These hard constraints maintain consistency by conditionally propagating binary values, which relate to the action state and collision avoidance of robots, to the reformulated continuous variables. The second approach is more aware of the branch-and-bound method which is the fundamental algorithm of modern MILP solvers. This approach is to guide the MILP solver to find integer solutions with shallower branching by adding a soft constraint, which softly restricts a robot's routes around delivery objects.

We demonstrated a considerable speed-up by the proposed models.

REFERENCES

[1] International Federation of Robotics, "World robotics report 2020," https://ifr.org/, 2020.

[2] 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," *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 639–646, 2014.

[3] T. L. Perez and P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 3684–3691, 2014.

[4] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "Incremental task and motion planning: a constraint-based approach," *Robotics: Science and Systems (RSS)*, 2016.

[5] C. Zhang and J. A. Shah, "Co-optimizing task and motion planning," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 4750–4756, 2016.

[6] M. Toussaint, "Logic-geometric programming: an optimization-based approach to combined task and motion planning," *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 1930–1936, 2015.

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

[8] M. Katayama, S. Tokuda, M. Yamakita, and H. Oyama, "Fast LTL-based flexible planning for dual-arm manipulation," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 6605–6612, 2020.

[9] R. Takano, H. Oyama, and M. Yamakita, "Continuous optimization-based task and motion planning with signal temporal logic specifications for sequential manipulation," *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2021.

[10] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, pp. 1–12, 2020.

[11] M. d. S. Atrantes, C. F. M. Toledo, B. C. Williams, and M. Ono, "Collision-free encoding for chance-constrained nonconvex path planning," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 433–448, 2019.

[12] R. Deits and R. Tedrake, "Efficient mixed-integer planning for UAVs in cluttered environments," *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015.

[13] H. Dai, G. Izatt, and R. Tedrake, "Global inverse kinematics via mixed-integer convex optimization," *Int. Journal of Robotics Research,*, vol. 38, no. 12–13, pp. 1420–1441, 2019.

[14] R. E. Bixby, "A brief history of linear and mixed-integer programming computation," *Documenta Mathematica: Optimization Stories*, pp. 107-121, 2012.

[15] Gurobi Optimization, LLC, "Gurobi 9.0 performance benchmarks," https://www.gurobi.com/wp-content/uploads/2020/02/Performance-Gurobi-9.0-1.pdf, 2020.

[16] IBM Corporation, "CPLEX optimization studio 12.10 performance improvements," https://www.ibm.com/downloads/cas/KEVDB4NZ, 2020.

[17] J. P. Vielma, "Mixed integer linear programming formulation techniques," *SIAM review*, Society for Industrial and Applied Mathematics, vol. 57, no. 1, pp. 3–57, 2015.

[18] J. A. Huchette, "Advanced mixed-integer programming formulations : methodology, computation, and application," Ph.D. Dissertation, Massachusetts Institute of Technology, 2018.

[19] T. A. M. Toffolo and H. G. Santos, "Python-mip," GitHub repository, https://github.com/coin-or/python-mip, 2021.

[20] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," https://www.gurobi.com/documentation/9.1/refman/index.html, 2021.