2025 IEEE/SICE International Symposium on System Integration (SII)
January 21-24, 2025, Munich, Germany

Task and Motion Planning of Fetch-and-Carry Including Push-Aside Action Using Mixed Integer Linear Programming

Yusuke Kuribayashi, Sotaro Sowa, Keisuke Takeshita, Kimitoshi Yamazaki, *Member, IEEE*

Abstract— **This paper describes a method for solving task planning and motion planning problems simultaneously. We target a fetch-and-carry of a small item by a single-arm mobile manipulator and introduce a method that can generate a sequence of actions and motions required for each action, even in environments with narrow open spaces such as corridors. In addition, to deal with a case where another object is already placed at the target location, we introduce a push-aside action and extend the previous method to include this action. As our method is formulated using Mixed Integer Linear Programming (MILP), the calculation time is relatively short, irrespective of the complexity of the target problem. To verify the efficiency of the proposed method, we performed a quantitative evaluation through simulation and conducted experiments on an actual mobile manipulator to verify feasibility of the methods.**

I. Introduction

Humans have both high maneuverability and mobility; therefore, they can perform a variety of tasks by combining these abilities. If we can artificially reproduce these abilities, we can promote automation in daily life support, industry, and services. From this perspective, mobile manipulation is being actively studied. Target tasks include door opening [1], cooperative transportation [2], furniture manipulation [3], etc. Among these, fetch-and-carry is positioned as a basic task. This task is simple but requires planning at several levels and has issues to be considered across several research fields.

The objective of this study is to show how to solve TAsk and Motion Planning (TAMP) problem for fetch-and-carry using a mobile manipulator. The goal is to simultaneously obtain a sequence of actions to solve a task and the robot's motions for each action. If the TAMP problem is properly solved in this way, the sequence of actions is more feasible than if task planning and motion planning are performed independently. On the other hand, mobile manipulators have multiple aspects, such as hand movement can be realized by either a mobile platform or a manipulator, which complicates the number of elements to be considered and their relationship. Furthermore, the authors try to include situations such as pushing any nontarget object aside when that object has been placed at the target location. In this case, the problem to be solved becomes even more complex, and solving the TAMP problem in a straightforward manner may require significant processing time.

The authors have previously expressed the TAMP problem of fetch-and-carry work by a mobile manipulator as a Mixed Integer Linear Programming (MILP) problem and have proposed a method to solve it by mathematical optimization [4]. MILP is a linear programming problem in which some variables take only integer values, and the solution can be obtained stably in a relatively short computation time. In this study, to further confirm the applicability of this method, we make the following extensions and verifications:

- We propose a MILP-oriented method that can generate both an action sequence and the motions required for each action for fetch-and-carry work, even in environments with narrow open spaces such as corridors.
- To deal with the case where a nontarget object is already placed at the target location, we introduce a push-aside action and extend the previous method to include this action in the MILP formulation.
- Using a single-arm mobile manipulator, we quantitatively evaluated the proposed method by simulation and confirmed its feasibility by experiments with an actual mobile manipulator.

This paper is structured as follows. Section II presents related work and Section III presents the problem setting and our approach. In Section IV, we explain the proposed method. Section V introduces our experimental results and discussion, and Section VI concludes this study.

II. Related Work

Many methods have been proposed to solve TAMP problem. Traditionally, the use of PDDL and extensions to symbolic planners have been considered to make TAMP easier to solve. One method is to use Hierarchical Task Networks (HTN). The method proposed in Wolfe et al. [5] reuses optimal solutions to state-abstracted subproblems across the search space. Silva et al. [6] proposed a method that introduces an intermediate layer between task planning and geometric reasoning. Kaelbling et al. [7] proposed a hierarchical method that makes choices and commits to TAMP in a top-down fashion to limit the length of plans that need to be constructed. Srivastava et al. [8] proposed a method that uses a novel representational abstraction and requires only those failures in computing a motion plan for a high-level action to be identifiable and expressible in the form of logical predicates at the task level. Zhang et al. [9] proposed a method for solving large-scale pick-and-place tasks by formulating them as traveling salesman problems. These hierarchical planning methods generally compute the task plan and motion plan alternately and separately. Therefore, the problem is to guarantee the solvability of each other.

Another approach has used mathematical optimization. Methods for solving TAMP as a constraint satisfaction

Y. Kuribayashi and S. Suwa is with Graduate School of Engineering, Shinshu University, Wakasato 4-17-1, Nagano, Nagano, Japan (corresponding author to provide e-mail: 23w4027a@shinshu-u.ac.jp).

K. Takeshita is with the Toyota Motor Cooperation.

K. Yamazaki is with the Faculty of Engineering, Shinshu University, Wakasato 4-17-1, Nagano, Nagano, Japan (corresponding author to provide e-mail: kyamazaki@shinshu-u.ac.jp).

problem (CSP) or formulating the problem as a nonlinear or linear optimization problem have been proposed. Lozano-Perez et al. [10] proposed a formulation for hierarchical problems in a discretized configuration space. The resulting problems can be solved using methods for discrete CSP. Lagriffoul et al. [11] introduced an intermediate layer between task planning and geometric reasoning. Toussaint et al. [12, 13] formulated a task that robots assist a human in picking objects into a nonlinear optimization problem and made planning possible. Hartman et al. [14] also used nonlinear optimization to plan robotic assembly tasks. Solving TAMP as an optimization problem like these has the advantage that task planning and motion planning can be computed simultaneously. However, the optimization problem is complicated by the representation of composite situations due to the presence of obstacles and the consideration of the robot's joint angles. This leads to a longer calculation time.

III. Problem Settings and Approach

A. Problem Settings

The task assumed in this study is to move a small object from one location to another. We refer to this object as a delivery object. The object is light and small enough to be grabbed with one hand. The environment in which the robot operates is a room or corridor. Impassable areas on which the delivery objects or other objects are placed, such as walls or tables, may be found in the environment. Also, when the goal position of a delivery object is given, there may be other items there. Such an item may be pushed aside; however, if there are walls or objects surrounding the item, the direction of the push will be limited. The initial and goal positions of a delivery object may be far from the robot's initial position.

The robot that performs the task is a mobile manipulator with a single articulated arm mounted on an autonomous mobile platform. The mobile platform can move in a straight line, rotate, or a combination of the two. The manipulator is assumed to have an articulated structure and a hand capable of grasping an object. The mobile manipulator is assumed to be able to grasp and transport only one item at a time.

Based on the above assumptions, the initial position of the robot, the initial position of each object to be delivered, their goal locations, and the shapes and arrangements of environment and obstacles are given in. For each of these inputs, the robot is asked to determine which deliveries to place, in what order, and where to place them, as well as the sequence of motions to be performed by the robot in each step of the sequence. Motion here consists of a combination of the trajectory of the mobile platform and the posture of the manipulator: when picking up a delivery, transporting and placing the delivery, and pushing other objects aside.

B. Approach

In setting up the problem, as described in the previous subsection, one approach would be to independently obtain the sequence of motions by planning at the task level and obtain each motion by planning at the motion level. However, in this case, it is difficult to guarantee mutual solvability. Therefore, in this study, we construct a method to simultaneously generate a motion sequence and each motion in a coordinated manner.

Recently, approaches that use mathematical optimization have been attracting considerable attention as a means of achieving this. The authors focus on MILP, a form of mathematical optimization, which is a linear programming problem in which some variables take integer values only and can be solved stably in a relatively short computation time. Kogo et al. [15] used the TAMP solution method with MILP to solve a linear programming problem of moving involving the movement of an object by a fixed manipulator. Further developing this method, the authors proposed a simultaneous TAMP solution method for mobile manipulators [4]. It was confirmed that the proposed method can compute a sequence of actions for placing multiple items in less than one second in almost all cases, with the aim of minimizing the movement path of a mobile manipulator and the number of steps involved in the work.

The MILP-oriented formulation described above is intended for simple fetch-and-carry tasks. However, in the real world, a robot may move around in a narrow space. Another object may be placed at the position where the robot wants to place the object currently grasped. In this case, we extend the method of [4] to handle such situations. Without sacrificing the advantages of the previous method, such as high computational speed, we have devised a new method for setting constraints that enables the addition of narrow-space traveling and push-aside action.

IV. MILP-Oriented Formulation for TAMP

A. The Overview of This Section

In this section, we first describe the methods proposed in the literature [4] in IV-A through D. We then describe an extended method for push-aside motion in IV-E, and we describe the concept for solving simultaneous TAMP in a confined space in IV-F. Note that the detailed constraints on the manipulator posture are not directly related to the proposal of this paper and can be found in reference [4].

In explaining the method, known information is denoted by upper-case letters, and other variables are denoted by lower case letters. The input information is as follows. The total number of phases is N_{pha}. The initial position of the robot is denoted by the real-valued vector $\boldsymbol{P}_{k=1}^{rob} = \left(X_k^{rob}, Y_k^{rob}\right)$, which represents the center position of the mobile platform in phase $k = 1$, where $k \in \{1, \cdots, N_{pha}\}$. The number of delivery objects is denoted by N_{dlv}. The initial and goal positions of the i-th delivery object are denoted by the real-valued vectors $\boldsymbol{P}_{i,k=1}^{dlv} = \left(X_{i,1}^{dlv}, Y_{i,1}^{dlv}, Z_{i,1}^{dlv}\right)$ and $\boldsymbol{P}_i^{goal} = \left(X_i^{goal}, Y_i^{goal}, Z_i^{goal}\right)$, respectively, where $i \in \{1, \cdots N_{dlv}\}$.

Now, we present the output information. The position of the mobile platform is denoted by $\boldsymbol{p}_k^{rob}$ and that of the end-effector $\boldsymbol{p}_k^{eef} = \left(x_k^{eef}, y_k^{eef}, z_k^{eef}\right)$. Although the robot's action is also an output, it is difficult for MILP to account for nonlinear factors, such as the trigonometric functions that is used for the orientation of the mobile platform and the joint angles of the manipulator. Therefore, the orientation of the mobile platform is set to finite patterns —forward, backward, left, and right—with the x-axis positive direction of the world coordinate system as the forward direction. Next, assuming that there are N_{pos} candidates that serve as the grasping/placing postures of the manipulator. Let $\theta_k^{fore,\alpha}, \theta_k^{back,\alpha}, \theta_k^{left,\alpha}$, and $\theta_k^{right,\alpha}$ represent orientations and postures of the robot. These variables $\theta_k^{*,\alpha}$ are with 0-1 variables in phase k, where $\alpha \in N_{pos}$. That is, there is a

$4 \times N_{pos}$ pattern of combinations of these variables, with one of them being 1. Meanwhile, the 0-1 variables that represent the robot action in each phase are denoted by $(\theta_{i,k}^{pck}, \theta_{i,k}^{cry}, \theta_{i,k}^{plc}, \theta_{i,k}^{gsp})$. For instance, $\theta_{i,k}^{pck}$ is 1 when the robot is picking up the i-th delivery object in phase k. Similarly, $\theta_{i,k}^{cry}$ is 1 when carrying, $\theta_{i,k}^{plc}$ is 1 when placing, and $\theta_{i,k}^{gsp}$ is 1 when grasping. In summary, multiple robot postures and actions are prepared in advance, and only their selection is performed during the optimization calculation.

B. Objective Function

The objective function is as follows:

$$\text{min.} \ -W_1 \sum_i \sum_k \delta_{i,k} + \sum_k (m_k^x + m_k^y) + W_2 \sum_k m_k^{xy}, \quad (1)$$

where W_1 and W_2 are weight coefficients. The first term, $\delta_{i,k} \in \{0, 1\}$, is a binary variable that takes the value 1 when the i-th delivery is completed. This term serves to reduce the number of phases in the entire operation. The second term is the sum of the moving distance of the mobile platform, where the distance in the x-direction in phase k is m_k^x and the distance in the y-direction in phase k is m_k^y. This term serves to minimize the total moving distance as a Manhattan distance. Then, the third term is added to search for the shortest moving distance at a Euclidean metric. This term makes it easier to select a diagonal move when multiple possible paths have the same Manhattan distance.

C. Constraints for Robot Action and Robot Posture

The constraint on the mobile platform is as follows:

$$-m_k^x \le x_k^{rob} - x_{k-1}^{rob} \le m_k^x, \quad (2)$$

where $x_k^{rob} (k > 1)$ is a real value that represents the x-coordinate of the center of the mobile platform. m_k^y is defined in the same way by using $y_k^{rob} (k > 1)$. The constraint on the third term of Eq (1), $m_k^{xy} (\ge 0)$, can be expressed as follows:

$$-m_k^{xy} \le m_k^x - m_k^y \le m_k^{xy}. \quad (3)$$

The constraints on the order of actions are as follows. For the simplicity of notation, we denote NOT, AND, and OR by $\neg$, $\wedge$, and $\vee$, respectively. The flow of grasping, carrying, placing, etc. when the robot performs a fetch-and-carry task is represented by switching 0-1 of $\theta_{i,k}^{gsp}$, which indicates whether the robot is grasping the delivery object or not. Based on this description rule, the following constraint formulas are defined:

$$\theta_{i,k}^{pck} = (\neg\theta_{i,k-1}^{gsp}) \wedge \theta_{i,k}^{gsp}, \quad (4)$$

$$\theta_{i,k}^{plc} = (\neg\theta_{i,k}^{gsp}) \wedge \theta_{i,k-1}^{gsp}, \quad (5)$$

$$\theta_{i,k}^{cry} = \theta_{i,k}^{gsp} - \theta_{i,k}^{pck}. \quad (6)$$

Eq. (4) indicates that $\theta_{i,k}^{pck}$ is set to 1 if the i-th delivery object is not grasped in phase $k-1$ and is grasped in phase k. Eq. (5) indicates the converse. Eq. (6) indicates that the robot holds the i-th delivery object in phase k and is not currently performing a grasping action: it is in the process of transporting the delivery object ($\theta_{i,k}^{cry} = 1$).

The following constraint is required to make the robot carry only one delivery object at a time:

$$\sum_i \theta_{i,k}^{gsp} \le 1. \quad (7)$$

The sum of $\theta_{i,k}^{gsp}$ of all deliveries in phase k should be 1.

When grasping, carrying, or placing a delivery object, the robot's end-effector position and the delivery position must be the same. This constraint is as follows:

$$-M(\neg\theta_{i,k}^*) \le x_{i,k}^{dlv} - x_k^{eef} \le M(\neg\theta_{i,k}^*). \quad (8)$$

where M is an arbitrary constant that is very large, and the * can represent cry, pck, or plc. This equation indicates that the values of $x_{i,k}^{dlv}$, which is the x-coordinate of the delivery object, and x_k^{eef}, which is the x-coordinate of the end effector, are the same when $\neg\theta_{i,k}^*$ is set to 0. The same can be said for the corresponding y-coordinate.

When the robot is not grasping the i-th delivery object, the following equation is used to indicate that the delivery object does not move.

$$-M\theta_{i,k}^{gsp} \le x_{i,k+1}^{dlv} - x_{i,k}^{dlv} \le M\theta_{i,k}^{gsp}. \quad (9)$$

This is the equation for the x-coordinate. The equation for the y-coordinate is represented in the same manner.

Next, the constraints for $\delta_{i,k}$ in Eq. (1) are as follows:

$$-M(1 - \delta_{i,k}) \le X_i^{goal} - x_{i,k}^{dlv} \le M(1 - \delta_{i,k}), \quad (10)$$

$$\delta_{i,k} \le \neg\theta_{i,k}^{gsp}, \quad (11)$$

$$\delta_{i,k+1} - \delta_{i,k} \ge \theta_{i,k}^{gsp} - \theta_{i,k+1}^{gsp}, \quad (12)$$

$$\delta_{i,1} \le \delta_{i,2} \le \cdots \le \delta_{i,N_{pha}} = 1. \quad (13)$$

Eq. (10) defines the constraint that $\delta_{i,k}$ can only take the value 1 if the x-coordinates of the i-th delivery object coincide with the position at which the object is to be placed in phase k. The same equations are also defined to represent whether the y- and z-coordinates of the current and goal positions coincide. Eq. (11) is the constraint that $\delta_{i,k}$ does not become 1 when the delivery object is held by the robot. Eq. (12) is the constraint that $\delta_{i,k+1}$ is 1 when the object is in a placed state after transitioning from phase k to phase $k+1$. Eq. (13) is the constraint that the delivery objects always arrive at the goal position in the last phase.

Constraints are also placed on the posture of the robot. Let us assume that the robot adopts a manipulator posture α. We define for $\theta_{i,k}^{fore,\alpha}, \theta_{i,k}^{back,\alpha}, \theta_{i,k}^{left,\alpha}, \theta_{i,k}^{right,\alpha}$ as follows:

$$\sum_\alpha (\theta_{i,k}^{fore,\alpha} + \theta_{i,k}^{back,\alpha} + \theta_{i,k}^{left,\alpha} + \theta_{i,k}^{right,\alpha}) = 1, \quad (14)$$

$$\theta_{i,k}^{fore,\alpha} + \theta_{i,k}^{back,\alpha} + \theta_{i,k}^{left,\alpha} + \theta_{i,k}^{right,\alpha} = \sum_i \theta_{i,k}^{\alpha}. \quad (15)$$

$$\sum_\alpha \theta_{i,k}^{\alpha} \le 1, \quad (16)$$

$$\sum_\alpha \theta_{i,k}^{\alpha} = \theta_{i,k}^{pck} + \theta_{i,k}^{plc}. \quad (17)$$

In Eq. (16), the sum of $\theta_{i,k}^{\alpha}$ cannot be greater than 1 and is 0 in phases other than when picking and placing. Also, Eq. (17) shows that the left side will not be 1, unless either $\theta_{i,k}^{pck}$ or $\theta_{i,k}^{plc}$ is 1.

D. Constraints for Collision Avoidance

The obstacles are approximated as rectangular and solid, and the mobile platform is considered rectangular on a

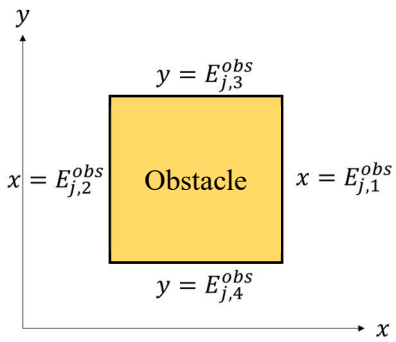


Fig. 1 Definition of obstacle edges

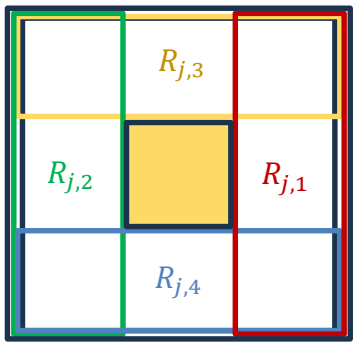


Fig. 2 Definition of existence region

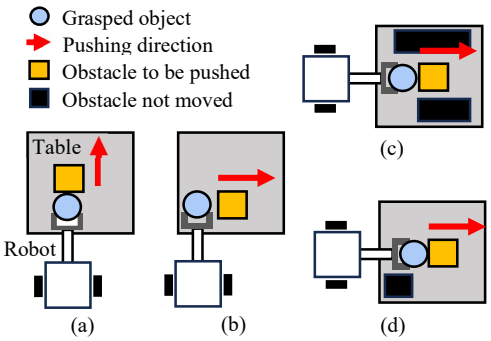


Fig. 3 Examples of pushing situation

horizontal plane. The number of obstacles is N_{obs}, and each obstacle is represented by four line segments $x = E_{j,1}^{obs}, E_{j,2}^{obs}$ $(E_{j,1}^{obs} > E_{j,2}^{obs})$, $y = E_{j,3}^{obs}, E_{j,4}^{obs}$ $(E_{j,3}^{obs} > E_{j,4}^{obs})$, as shown in Fig. 1. Here, j means the j-th obstacle, where $j \in \{1, \cdots, N_{obs}\}$. To define an obstacle area, four duplicated regions are set as shown in Fig. 2. Note that moving to a different region $R_{j,n}$ in one move will result in crossing an obstacle. Considering this, the following constraints are used:

$$\bigvee_{n=1,2,3,4} \left(\boldsymbol{p}_k^{rob} \in R_{j,n}\right) \wedge \left(\boldsymbol{p}_{k+1}^{rob} \in R_{j,n}\right) = 1. \quad (18)$$

This constraint requires that the robot be in the same non-collision region $R_{j,n}$ during both phase k and phase $k+1$, where $k \in \{1, \cdots, N_{pha}\}$.

If an obstacle is on a table, constraints must be applied to prevent the mobile platform and manipulator from interfering with it. This constraint is expressed as follows:

$$\bigvee_{n=1,2,3,4} \left(\boldsymbol{p}_k^{eef} \in R_{j,n}\right) \wedge \left(\boldsymbol{p}_k^{rob} \in R_{j,n}\right) = 1. \quad (19)$$

E. Constraints for Pushing an Obstacle Aside

If an obstacle is present at the goal position of a delivery, it can be removed from the location by pushing it away from the position. However, it is not desirable for the pushed obstacle to fall off the table. Also, the direction of pushing should be determined by the position of other obstacles on the table. In this study, the obstacle to be pushed should not collide with other obstacles in order to avoid complicating the placement process. Fig. 3 shows some patterns of pushing directions. Basically, only two ways of pushing are allowed: pushing forward, as shown in Fig. 3(a), or pushing with lateral movement, as shown in Fig. 3(b). In Fig. 3(c), the push should be in the forward direction because pushing in the lateral direction will cause obstacles to interfere with each other. In Fig. 3(d), the robot should push forward because moving laterally will cause interference between the robot and the obstacles. The following is a description of the method used to select the pushing pattern properly.

First, preprocessing of optimization is set as follows. Let u_i be a 0-1 variable, and set $u_i = 1$ when an obstacle already exists at the goal position $\boldsymbol{P}_i^{goal}$ where the i-th object should be placed on. Let the edges of the obstacle to be pushed be $\boldsymbol{E}_i^{pobs} = (E_{i,1}^{pobs}, E_{i,2}^{pobs}, E_{i,3}^{pobs}, E_{i,4}^{pobs})$. Make the following settings if $u_i = 1$:

$$\left(E_{i,1}^{plctbl} - E_{j,1}^{obs} > 0\right) \wedge \left(-E_{i,1}^{pobs} + E_{j,2}^{obs} > 0\right) \Leftrightarrow s_{i,1} = 1,$$
$$\left(-E_{i,2}^{plctbl} + E_{j,2}^{obs} > 0\right) \wedge \left(E_{i,2}^{pobs} - E_{j,1}^{obs} > 0\right) \Leftrightarrow s_{i,2} = 1,$$
$$\left(E_{i,3}^{plctbl} - E_{j,3}^{obs} > 0\right) \wedge \left(-E_{i,3}^{pobs} + E_{j,4}^{obs} > 0\right) \Leftrightarrow s_{i,3} = 1,$$
$$\left(-E_{i,4}^{plctbl} + E_{j,4}^{obs} > 0\right) \wedge \left(E_{i,4}^{pobs} - E_{j,3}^{obs} > 0\right) \Leftrightarrow s_{i,4} = 1,$$

where $s_{i,n}$ is a 0-1 variable. When $s_{i,n} = 1$, the above equations indicate that other obstacles exist in the inner area R_n of the obstacle to be pushed. Next, make the following settings if $s_{i,n} = 1$:

$$\left(-E_{i,3}^{puobs} + E_{j,3}^{obs} > 0\right) \wedge \left(E_{i,4}^{puobs} - E_{j,4}^{obs} > 0\right) \Leftrightarrow o_{i,1} = 1,$$
$$\left(-E_{i,3}^{puobs} + E_{j,3}^{obs} > 0\right) \wedge \left(E_{i,4}^{puobs} - E_{j,4}^{obs} > 0\right) \Leftrightarrow o_{i,2} = 1,$$
$$\left(-E_{i,1}^{puobs} + E_{j,1}^{obs} > 0\right) \wedge \left(E_{i,2}^{puobs} - E_{j,2}^{obs} > 0\right) \Leftrightarrow o_{i,3} = 1,$$
$$\left(-E_{i,1}^{puobs} + E_{j,1}^{obs} > 0\right) \wedge \left(E_{i,2}^{puobs} - E_{j,2}^{obs} > 0\right) \Leftrightarrow o_{i,4} = 1,$$

where $o_{i,n}$ is a 0-1 variable. When $o_{i,n} = 1$, the above equations indicate that collision occurs with other obstacles when the obstacle to be pushed is pushed to area R_n.

Therefore, pushing an obstacle in the lateral direction is prohibited in the above cases. The variables used to judge it are as follows:

$$s_{i,1} + o_{i,1} + s_{i,2} + o_{i,2} \geq 3 \Leftrightarrow q_i^{R_{1,2}} = 1,$$
$$s_{i,3} + o_{i,3} + s_{i,4} + o_{i,4} \geq 3 \Leftrightarrow q_i^{R_{3,4}} = 1,$$
$$(u_i = 1) \wedge \left(o_{i,1} + o_{i,2} + o_{i,3} + o_{i,4} = 0\right) \Leftrightarrow q_i^{(4)} = 1,$$

where $q_i^{R_{*,**}}$, $q_i^{(4)}$ are 0-1 variables. The first two of the above equations indicate that $q_i^{R_{*,**}}$ is set to 1 when other obstacles exist in the inner regions R_*, R_{**} of the items to be removed. In the third equation, $u_i = 1$ but $o_{i,1} + o_{i,2} + o_{i,3} + o_{i,4} = 0$ means that $q_i^{(4)}$ is set to 1 when the other obstacles are in one of the four diagonal corners from the obstacle to be pushed (see Fig.3 (d)). The above is the preprocessing for the optimization.

For the optimization calculation, the following constraints are added to generate a trajectory for pushing when an obstacle to be pushed is present.

$$u_i \leq m_k^x + m_k^y \leq M - (M + B) \cdot \theta_{i,k}^{plc} \cdot u_i, \quad (20)$$

where B is a constant and indicates the amount of motion to push aside. The following constraints are added to prevent a situation in which a pushed obstacle collides with other obstacles:

$$0 \leq m_k^x \leq M + u_i \cdot \theta_{i,k}^{plc} F_{1,2}, \quad (21)$$
$$0 \leq m_k^y \leq M + u_i \cdot \theta_{i,k}^{plc} F_{3,4}, \quad (22)$$

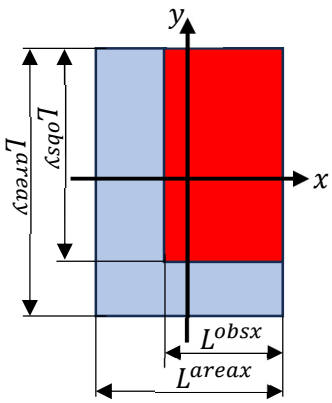


Fig. 4 Area definition for narrow space

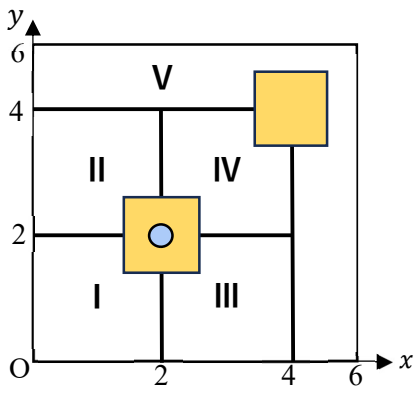


Fig. 5 Area definition at verification simulation

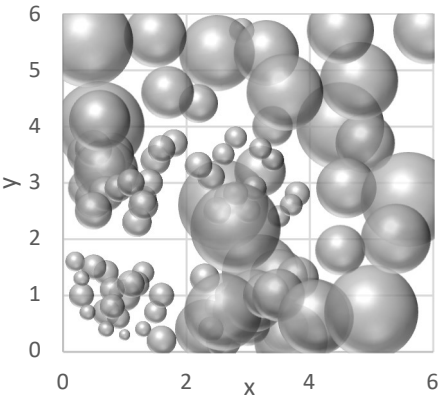


Fig. 6 Obstacle position vs. computation time

where

$$F_{a,b} = -(M - B) - B\{1 - (1 - o_{i,a})(1 - o_{i,b})\}.$$

In Eqs. (21) and (22), $o_{i,n}$ is set to 1 if the obstacle to be pushed collides with other obstacles as a result of the pushing action. Therefore, when both u_i and $\theta_{i,k}^{plc}$ are set to 1, the amount of push in the direction of the other objects becomes 0. Finally, either $q_i^{R_{1,2}}, q_i^{R_{3,4}}$, or $q_i^{(4)}$ equals 1, then the following constraints are added:

$$-M - G_{3,4,up} \le x_{k+1}^{rob} - x_k^{rob} \le M + G_{3,4,down}, \quad (23)$$
$$-M - G_{1,2,left} \le y_{k+1}^{rob} - y_k^{rob} \le M + G_{1,2,righ}, \quad (24)$$

where

$$G_{a,b,c} = \left(q_i^{R_{a,b}} + q_i^{(4)}\right)\left(1 - \theta_{i,k+1}^{plc} - \sum_{\alpha} \theta_{i,k+1}^{c,\alpha}\right)(M + B).$$

In Eqs. (23) and (24), G is used to calculate the amount of push: set B if pushing action is needed: otherwise, set it to 0. Note that these constraints are not limiting factors in the ranges of x and y unless pushing action is required.

The constraints to prevent diagonal pushing are as follows.

$$u_i\theta_{d,t}^{plc} \le h_k^x + h_k^y \le 2 - u_i\theta_{i,k}^{plc}, \quad (25)$$
$$0 \le m_k^x \le M(2 - h_k^x - \theta_{i,k}^{plc}), \quad (26)$$

where h_k^x and h_k^y are 0-1 variables to limit the amount of motion of the robot in the x- and y- directions. Eq. (25) is a constraint that either h_k^x or h_k^y be 1 in the placing phase if an obstacle to be pushed is present. Eq. (26) is a constraint that the robot does not move in the x-direction when it is in the placing phase and h_k^x is 1. The same constraint is also defined for the y-direction.

F. Definition for TAMP at Narrow Space

As a development from previous work, we introduce a method for planning in a narrow space. The general policy is to define a rectangular area that includes the space in which the robot can move, and then to define an impenetrable area within the area as an obstacle. Let us take the L-shaped corridor shown in Fig. 4 as an example. We define a rectangle $L^{are} \times L^{areay}$ that encompasses the corridor area, and we add the following constraints, with the center of the area as the reference position $(x, y) = (0,0)$.

$$-\frac{1}{2}L^{areax} \le x_k^{rob} \le \frac{1}{2}L^{areax} \quad (27)$$

A similar constraint is added for y_k^{rob}, with L^{areay} as the length of the room in the y-direction. Then, the part of the room that is not a corridor (the red part in Fig. 4) is considered an obstacle, and the constraints are set according to the way described in Section IV-D.

V. Verification

A. Settings

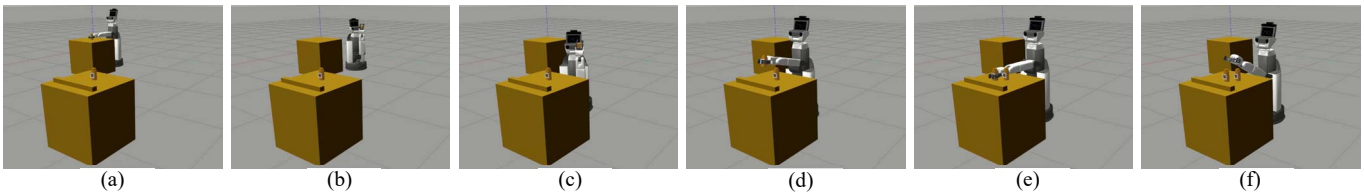
The feasibility of the proposed method for a fetch-and-carry task was verified. The delivery object was assumed to be cylindrical object (e.g., a plastic bottle) and could be grasped from any direction. A gazebo was used as a physics simulator. The robot used was a human support robot (HSR) [16], which is manufactured by Toyota Motor Corporation. The robot has eight Degrees of Freedom (DoFs), including a mobile platform having three DoFs and a manipulator with five DoFs

Two postures are given as α. One is the posture, where the end effector is extended far from the table edges, and the other is the posture used to carry out work in front of the table. Other settings for the optimization include starting with $N_{pha} = 3$ and increasing this value by 1 if a solution cannot be derived. The reason for this is that setting a large N_{pha} under conditions where a small N_{pha} would be sufficient to complete the task consumes extra execution time. SCIP was used as a solver.

B. Basic Performance

To investigate the computation time and performance of the solutions, the following simulation was conducted. The initial position and goal position of deliveries were set to $(X_{i,1}^{dlv}, Y_{i,1}^{dlv}) = (2,2)$ and $(X_i^{goal}, Y_i^{goal}) = (4,4)$, respectively, and a table 0.6 m square in size was placed at each position. Under these conditions, at least one avoidance moving path was required. In addition, one obstacle was placed an area of 0.4 m square at a random location.

To statistically evaluate the proposed method, the obstacle placement area was divided into five, as shown in Fig. 5. The computation times [s] with respect to each area were as follows (stated for "mean (SD)"). Area I: 0.05 (0.02), II: 0.13 (0.09), III: 0.22 (0.20), IV: 0.19 (0.23), V: 0.36 (0.23). Fig. 6 shows the resulting bubble chart. The position of the bubble indicates the position of the obstacle, and its size indicates the ratio of the computation time.



(a) (b) (c) (d) (e) (f)

Fig. 7 Example of pushing away an obstacle for a fetch-and-carry task

(a) (b) (c) (d) (e) (f)

Fig. 8 Fetch-and-carry execution by an actual HSR

Table I Obstacle placement pattern

	A	B	C	D	E
Obs 1	1	0	2	0	1
Obs 2	0	1	0	2	1

The presence of obstacles on the shortest path tended to narrow down the choices of the path that could be taken by the mobile platform and shorten the processing time. The solution was obtained without any problem, even for TAMP in a narrow environment; for example, an L-shaped corridor as shown in Fig. 4, required a computation time of 0.2 s.

C. Solving the TAMP Problem by Pushing Aside

Obstacles were placed on a table in the combinations shown in Table I, and a fetch-and-carry plan with push-aside was executed for each. One of the obstacles (Obs 1) had a long and wide shape that prevented pushing action in the direction in which it was placed, while another obstacle (Obs 2) was a small rectangular shape. For each combination, five TAMPs were executed with different obstacle locations, and the mean and standard deviations of the computation times [s] were measured. The results were as follows (stated for "mean (SD)"): A: 0.29 (0.18), B: 0.53 (0.08), C: 0.46 (0.15), D: 0.64 (0.19), and E: 0.55 (0.23). Fig. 7 shows an example of the physics simulation. Fig. 8 shows an example of an experiment using an actual robot. By using self-positioning combined with odometry and a laser rangefinder, it was possible to accomplish the same task in the simulation.

VI. Conclusion

In this paper, we present a simultaneous TAMP method for fetch-and-carry work by a mobile manipulator. We present a method that can generate a sequence of actions and motions required for each action, even in environments with narrow open spaces, such as corridors. In addition, to deal with a case where another object is already placed at the goal location, we proposed a push-aside action, and as a result, we extend the previous method to include this action. Using HSR, a mobile manipulator, we confirmed that our method enables the robot to execute manipulation work in the situation described above.

Future work includes finding ways to simplify the formulation and speed up calculations.

References

[1] Ito et al., "Efficient multitask learning with an embodied predictive model for door opening and entry with whole-body control," Science Robotics, Vol. 7, No. 65, 2022.

[2] A. Petitti et al., "Decentralized Motion Control for Cooperative Manipulation with a Team of Networked Mobile Manipulators," IEEE Int. Conf. on Robotics and Automation, pp. 441–446, 2016.

[3] K. Yamazaki et al., "Home Assistant Robot for an Aging Society," in Proc. of the IEEE, Vol. 100, No. 8, pp. 2429–2441, 2012.

[4] S. Suwa et al., "Task and Motion Planning Using Mixed Integer Linear Programming for Solving Fetch-and-Carry Tasks by a Mobile Manipulator," Advanced Robotics, Vol. 38, No. 17, pp. 1188–1201, 2024.

[5] J. Wolfe, B. Marthi, and S. J. Russell, "Combined task and motion planning for mobile manipulation." in ICAPS, pp. 254–258, 2010.

[6] L. Silva, et al., "Towards combining HTN planning and geometric task planning," arXiv preprint arXiv:1307.1482, 2013.

[7] L. P. Kaelbling and T. Lozano-Perez, "Hierarchical task and motion planning in the now," IEEE Int. Conf. on Robotics and Automation, pp. 1470–1477, 2011.

[8] 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, pp. 639–646, 2014.

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

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

[11] F. Lagriffoul et al., "Efficiently combining task and motion planning using geometric constraints." The Int. Journal of Robotics Research, Vol. 33, No. 14, 2014.

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

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

[14] V. N. Hartmann, O. S. Oguz, D. Driess, M. Toussaint, and A. Menges, "Robust task and motion planning for long-horizon architectural construction planning," IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2020.

[15] T. Kogo, K. Takaya, and H. Oyama, "Fast MILP-based Task and Motion Planning for Pick-and-Place with Hard/Soft Constraints of Collision-Free Route", IEEE Conf. on SMC, 2021.

[16] T. Yamamoto et al., "Development of the Research Platform of a Domestic Mobile Manipulator Utilized for International Competition and Field Test," IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 7675–7682, 2018.