Differentiable Task Assignment and Motion Planning

Jimmy Envall[1], Roi Poranne[2], Stelian Coros[1]

***Abstract*— Task and motion planning is one of the key problems in robotics today. It is often formulated as a discrete task allocation problem combined with continuous motion planning. Many existing approaches to TAMP involve explicit descriptions of task primitives that cause discrete changes in the kinematic relationship between the actor and the objects. In this work we propose an alternative, fully differentiable approach which supports a large number of TAMP problem instances. Rather than explicitly enumerating task primitives, actions are instead represented implicitly as part of the solution to a nonlinear optimization problem. We focus on decision making for robotic manipulators, specifically for pick and place tasks, and explore the efficacy of the model through a number of simulated experiments including multiple robots, objects and interactions with the environment. We also show several possible extensions.**

I. Introduction

To perform manipulation tasks, even simple ones such as picking up an object, robots must solve two tightly coupled problems. The first is decision making: what strategy should be used to pick the object? The second is motion planning: how to move and perform the picking motion in an optimal way based on the selected strategy? These problems together are a simplistic instance of the *task and motion planning* problem, or TAMP. Another instance of the problem involves several robotic manipulators that are tasked with sorting and organizing a set of objects that are scattered around their environment. One way to solve this problem is to assign each robot to specific pick-and-place tasks, and then find the optimal trajectories for all of them simultaneously. The main question TAMP aims to address is, what would be the optimal assignment? The challenge there stems from the interplay between the two problems that make up TAMP: task planning, and motion planning.

The cost of a motion plan given a specific task is hard to predict and expensive to evaluate, and even the smallest change to the task description can cause the motion plan to become infeasible. Thus, TAMP approaches are often concerned with finding efficient ways for searching in the space of task assignments [1], [2], [3].

The combinatorial complexity of TAMP could be partly mitigated if task planning could be formulated in a unified, continuous way. Furthermore, and perhaps more importantly, a continuous formulation would allow the problem to be

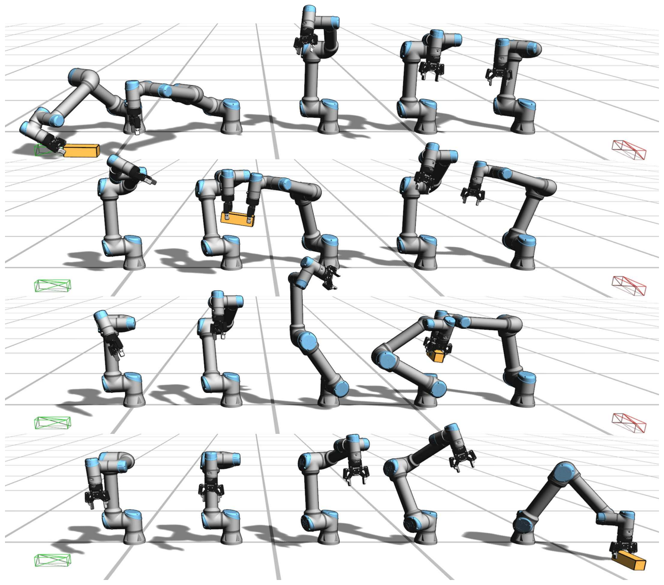


Fig. 1: The number of available actions, such as handovers, increases exponentially with the number of manipulators.

integrated in differentiable simulators and neural networks and ultimately be a step toward high-level decision making in complex situations. For these reasons, our goal in this paper is to propose a fully continuous formulation applicable to a considerable subset of the TAMP problem domain including task assignment and motion planning. Our emphasis is on decision-making for robotic manipulators, particularly for pick-and-place tasks. We address this challenge by treating task assignment in an *implicit* manner. The idea is, instead of assigning a pick-and-place task, to associate robots with objects using time-dependent, real functions. These functions, in some sense, express for each point in time a degree of which a specific robot should hold a specific object. Hence, in contrast to explicitly defined pick events and place events, they only *implicitly* define them.

Based on this concept, we define a smooth optimization problem that can be readily solved using gradient-based methods, such as Newton's method. We demonstrate the potential of this approach in a variety of settings, including multiple manipulators and objects. We also show that this approach is easily extendable to handle different grasps, handovers and more complex interactions.

II. Related work

Modern approaches for task and motion planning combine discrete high level decision making with continuous parameter search in different ways. In [1] the configuration space is discretized and combined with the action skeleton into a

[1] The authors are with the Department of Computer Science, ETH, Zurich, Switzerland. jimmy.envall@inf.ethz.ch; scoros@inf.ethz.ch

[2]Roi Poranne is with the Department of Computer Science, University of Haifa, Haifa, Israel. roiporanne@cs.haifa.ac.il

This work was supported by a ETH RobotX research grant funded through the ETH Zurich Foundation.

discrete constraint satisfaction problem that is then handed to a generic CSP solver. Another popular method known as Logic Geometric Programming (LGP) [2], [3] combines symbolic action search with a non-linear program for finding the trajectory. The exponential growth of the discrete action space poses challenges for longer planning horizons, for which several extensions have been developed [4], [5].

Other approaches, e.g. [6], [7] suggest to find the trajectory by sampling. In [6] the symbolic actions and the continuous parameter spaces are fused and solved simultaneously using an off-the-shelf motion planner while in [7] the domain specific constraints are utilized for factoring the problem, enabling efficient sampling. As can be inferred by [8], the main differences between different approaches are how they solve for the continuous variables and how constraint satisfaction interacts with the search for high level action sequences. In general, TAMP is a very active research topic, and the review in [8] is highly recommended.

TAMP covers a wide range of problems [9], and successful completion of a task may hinge on effective manipulation, which is an active research field on its own [10], [11], [12]. However, while task planning for a single manipulator already is challenging, increasing the number of agents puts additional emphasis on the planning and decision making algorithms. In some settings, e.g. in a factory or in a warehouse, less general algorithms may be sufficient. Pick and place problems involving multiple robots and handovers may be solved using heuristics and sampling for exploring the search space [13]. Another option might be to consider the individual robots in a non-cooperative fashion [14].

Planning through kinematic modes using optimization has previously been studied in [15] and [16] in the context of motion synthesis for computer graphics and in [17] for trajectory planning of rigid body systems. Similarly to our work, these contributions also rely on optimization and additional variables for describing switches between different modes. Sampling based methods include e.g. [18] which extends the popular probabilistic roadmaps method [19] with multi-modal capabilities.

A similar concurrent work found in [20] builds upon a concept known as signal temporal logic (STL) and a smooth approximation thereof [21], which enables a fully continuous approximation to task planning. The formulation proposed in [20] also features auxiliary variables similar to the ones used in this work. However, the formulation relies on constraints that scale quadratically with the number of objects. Additionally, it was not demonstrated on multiple robots, and does not include a grasping parametrization and uses floating end effectors during the planning with joint angles computed in a separate step.

The idea of treating naturally discrete phenomena as continuous has also been employed in other fields, such as topology optimization. A popular method for optimizing structural integrity is known as Solid Isotropic Micro-Structure with Penalization, or SIMP [22], [23]. In SIMP the target domain is divided into finite elements whose occupancy is modeled using a continuous range of values.

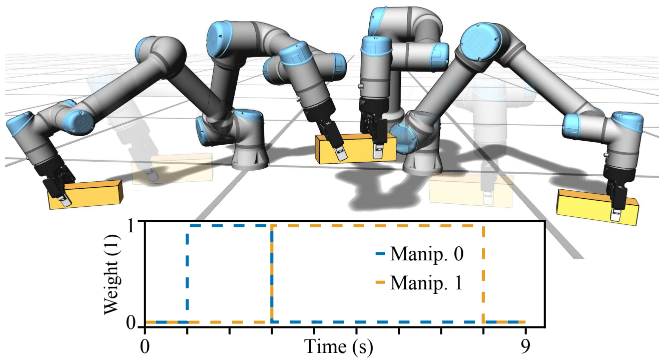


Fig. 2: Example setup with two robotic manipulators and one movable object.

For non-porous materials fractional values are difficult to interpret, and therefore penalization techniques are needed to drive the occupancy to either empty or solid. However, in our method, fractional values do not constitute an issue.

III. Overview

The archetypal trajectory optimization problem we build upon is

$$\begin{aligned} \min_{\mathbf{x}(t)} \quad & \mathcal{J}(\mathbf{x}(t)) \\ \text{s.t.} \quad & c(\mathbf{x}(t)) = 0, \end{aligned} \tag{1}$$

where $\mathbf{x}(t)$ denotes the trajectory of all state variables, $\mathcal{J}$ is the objective, e.g. shortest path, and $c(\mathbf{x})$ denotes user-specified constraints such as initial and target positions on the trajectory. To begin our discussion, consider a scenario where a robotic manipulator is holding a box. The final desired placement of the box is out of reach for the robot that is currently holding it, but there is another robot close by that can take the box from the first robot and bring it to its goal. This handover maneuver is a decisive high-level task that needs to be actively chosen. An illustration is shown in 2.

The pose of the object can be seen as a link in the kinematic chain. The pose is thus governed by the kinematic equations that are induced by the robot holding it. Put differently, there is a set of constraints that the box and the robot holding it must satisfy. During a handover, this set of constraints change in what is known as a *kinematic switch*.

Much of the previous work mentioned above deals with the explicit sequencing of actions. We instead enable the kinematic switches to emerge as a result of solving a nonlinear program. This is made possible by mollifying the switches. This way we can simultaneously find the action sequence and the motion trajectory using only gradient-based methods without relying on integer-based techniques.

This example can be expressed as an extension of (1):

$$\begin{aligned} \min_{\mathbf{x}(t)} \quad & \mathcal{J}(\mathbf{x}(t)) \\ \text{s.t.} \quad & c(\mathbf{x}(t)) = 0 \\ & c_1(\mathbf{x}(t)) = 0 \vee c_2(\mathbf{x}(t)) = 0 \vee c_\emptyset(\mathbf{x}(t)) = 0 \; \forall \, t \end{aligned}$$

where $\mathbf{x}(t)$ also includes the trajectory of the object. The constraints $c_1(\mathbf{x})$ and $c_2(\mathbf{x})$ define the grasping relationship between the object and each manipulator, and $c_\emptyset$ is a function whose value is zero when the object rests on the ground. The distinguishing feature of this problem is that only one of these constraints needs to be satisfied at a given time. In other words, the box must be held by one or both manipulators and/or be on the ground, but it cannot float.

While this could potentially be formulated as a mixed integer problem, we instead opt for a fully continuous formulation which we detail in the next section. Our main idea is to *relax* the kinematic switches in the problem by introducing time-dependent *association* weights $w_i(t) : [0, T] \to [0, 1], \;\; i \in \{1, 2\}$. These weights signify the association between each robot and the box, or "how much" the constraints c_1 and c_2 need to be satisfied. With these weights, the optimization problem is transformed into a complementarity problem:

$$\begin{aligned} \min_{\mathbf{x}, w_1, w_2} \quad & \mathcal{J}(\mathbf{x}(t)) \\ \text{s.t.} \quad & c(\mathbf{x}(t)) = 0 \\ & w_i(t) c_i(\mathbf{x}(t)) = 0, \forall\, i \in \{1, 2\} \\ & w_\emptyset(t) c_\emptyset(\mathbf{x}(t)) = 0 \end{aligned} \tag{2}$$

where $w_\emptyset(t) = 1 - \sum_i w_i(t)$. Any of the constraints c_1, c_2 and $c_\emptyset$ can now become inactive by choosing the weights w_1 and w_2 appropriately.

IV. Method

The problem presented in (2) provides the foundation of our approach. In this section we refine the formulation to add support for task allocation, multiple objects and robotic manipulators, and more.

Trajectories We denote the joint angles of a manipulator $i \in \mathcal{M}$ at time $t \in [0, T]$ by $\mathbf{m}_i(t) \in \mathbb{R}^n$. The joint angles can be subjected to box constraints of the form $\mathbf{m}_i^{\text{min}} \le \mathbf{m}_i(t) \le \mathbf{m}_i^{\text{max}}$ that correspond to the physical joint limits of the robot. The trajectory of an object $j \in \mathcal{P}$ is written analogously as $\mathbf{p}_j(t) \in \mathbb{R}^6$. Objects consist of only one rigid body and $\mathbf{p}_j$ thus directly describes an object's pose in world coordinates. We use $\mathbf{f} : \mathbb{R}^n \to \text{SE}(3)$ for denoting the forward kinematic function that maps the manipulator state $\mathbf{m}_i$ to an end effector position and orientation in the world coordinate frame.

Task selection The allocation of a task j to a manipulator i is modeled using a function $w_{ij}(t) : [0, T] \to [0, 1]$ as outlined in III. These variables form a key component of the path constraints that will be developed in IV-A.

Resting Objects may need to be released before they have reached the goal pose. To ensure physically plausible behavior we restrict the resting pose of the object to be in a set of stable configurations. E.g. a cube can be placed with any face towards the ground, implying that the elevation of the cube as well as one rotational axis is fixed. We introduce the functions $r_{dj}(t) : [0, T] \to [0, 1]$ where d corresponds to a specific resting pose. We use these functions to construct the resting orientation constraints in IV-B.

Grasping During handovers from one robot to another, different grasping poses may be necessary in order to prevent robots from colliding. In this work we use cuboid objects of size 0.06 m$\times$0.06 m$\times$0.2 m. We model the grasp pose with two degrees of freedom; one along the longitudinal axis of the object and one being the angle between the gripper and the local y-axis of the object. A similar parametrization has been used in [24] with an additional degree of freedom.

We denote the longitudinal offset and the angle with δ_{ij} and θ_{ij} respectively. The longitudinal offset is subject to a box constraint that depends on the physical dimensions of the object, i.e. $\delta_j^{\text{min}} \le \delta_{ij} \le \delta_j^{\text{max}}$. We note that the grasping parameters are defined per manipulator-object pair and therefore are independent of t.

For conciseness we hereafter write $\mathbf{y}_{ij}(t) = \begin{bmatrix} w_{ij}(t) & \delta_{ij} & \theta_{ij} \end{bmatrix}^T$. Given the grasping parameters and the object state $\mathbf{p}_j$ we can define $\mathbf{g} : (\mathbf{p}_j, \mathbf{y}_{ij}) \to \text{SE}(3)$ that maps the object state and grasping parameters to a grasping pose for the end effector in world coordinates.

A. Differentiable kinematic switches

We model the kinematic switches as path constraints that depend on $\mathbf{m}$, $\mathbf{p}$ as well as the auxiliary functions $w_{ij}(t)$. The constraint has the form

$$C_{\text{pos}} := w_{ij} \left(\mathbf{f}_i(\mathbf{m}_i) - \mathbf{g}(\mathbf{p}_j, \mathbf{y}_{ij}) \right) = \mathbf{0} \tag{3}$$

for manipulator i and object j. In addition we constrain the velocities of the non-actuated bodies such that they equal the weighted sum of the velocities of the end effectors that are currently moving it. The velocity constraint is of the form

$$C_{\text{vel}} := \dot{\mathbf{g}}(\mathbf{p}_j, \mathbf{y}_{ij}) - \sum_{i \in \mathcal{M}} w_{ij} \dot{\mathbf{f}}_i(\mathbf{m}_i) = \mathbf{0}. \tag{4}$$

This constraint also ensures that the velocity of an object is zero when $w_{ij} = 0 \; \forall \; i \in \mathcal{M}$. Together with (3), (4) ensures that an object will be moving only when a manipulator holds it. These two constraints thus fully describe the dynamical relationship between the manipulator and the objects to be manipulated.

B. Resting constraints

For a cube or a block with six faces the resting constraint can be expressed as a constraint on the elevation and on one of the orientation axes of the object. The number of constraints thus equals the number of available resting orientations. For brevity we write $\hat{w}_{dj}(t) = 1 - \sum_{i \in \mathcal{M}} w_{ij}(t)$. The resting constraint can then be expressed as

$$C_{\text{rest}} := \hat{w}(t)_{dj} r_{dj}(t) \phi_d(\mathbf{g}(\mathbf{p}_j, \mathbf{y}_{ij})) = \mathbf{0}$$

where ϕ_d is a function that captures the difference between the relevant rotation axis for the resting pose d as well as the resting elevation, e.g. corresponding to a floor or a table. By setting $\sum_d r_{dj}(t) = 1$ we ensure that at least one constraint is active when $\hat{w}(t)_{dj} > 0$.

C. Collisions

Collision free trajectories are ensured by imposing a collision constraint of the form

$$C_{\text{collision}} := c_{\text{collision}}(\mathcal{K}_a(\mathbf{m}, \mathbf{p}), \mathcal{K}_b(\mathbf{m}, \mathbf{p})) \ge 0 \quad \forall \quad a, b \in \mathcal{C}$$

where $\mathcal{K}_a, \mathcal{K}_b$ denote a pair of forward kinematic functions for collision primitives $a, b \in \mathcal{C}$ in the scene and the value of $c_{\text{collision}}$ is proportional to the squared distance between the collision primitives and $\mathbf{m}, \mathbf{p}$ denote the stacked trajectory vectors. We refer the reader to [25] for details.

D. Trajectory optimization problem

By stacking the manipulator and object trajectories as well as the grasping parameters into $\mathbf{m}$, $\mathbf{p}$ and $\mathbf{y}$ respectively we can now write the trajectory optimization problem as

$$\begin{aligned} \min_{\mathbf{m},\mathbf{p},\mathbf{y}} \quad & \mathcal{J}(\mathbf{m},\mathbf{p},\mathbf{y}) \\ \text{s.t.} \quad & C_{\text{pos}} = 0 && \forall\, i \in \mathcal{M},\ j \in \mathcal{P} \\ & C_{\text{vel}} = 0 && \forall\, j \in \mathcal{P} \\ & C_{\text{rest}} = 0 && \forall\, j \in \mathcal{P}, \\ & C_{\text{collision}} \geq 0 && \forall\, a, b \in \mathcal{C} \\ & \mathbf{m}_i^{\text{min}} \leq \mathbf{m}_i \leq \mathbf{m}_i^{\text{max}} && \forall\, i \in \mathcal{M} \\ & \delta_j^{\text{min}} \leq \delta_{ij} \leq \delta_j^{\text{max}} && \forall\, i \in \mathcal{M},\ j \in \mathcal{O} \end{aligned} \tag{5}$$

where $\mathcal{J}$ is an objective function. In the experiments we also penalize joint velocities and accelerations of the manipulators, i.e.

$$f_v(\mathbf{m}_i) := \|\dot{\mathbf{m}}_i\|^2$$

and

$$f_a(\mathbf{m}_i) := \|\ddot{\mathbf{m}}_i\|^2$$

as well as the end effector velocities

$$f_{\text{ee}}(\mathbf{m}_i) := \|\dot{\mathbf{f}}_i(\mathbf{m}_i)\|^2.$$

$\mathcal{J}$ then reads

$$\mathcal{J} = \sum_{i \in \mathcal{M}} \left(\beta_1 f_v(\mathbf{m}_i) + \beta_2 f_a(\mathbf{m}_i) + \beta_3 f_{\text{ee}}(\mathbf{m}_i)\right)$$

where $\beta_l \in \mathbb{R}, l \in \{1,2,3\}$ are constant weights.

The constraints in (5) are sufficient for preventing a single manipulator from holding multiple objects simultaneously. However, in our numerical experiments we have found that adding an additional constraint that limits the capacity of the manipulators is beneficial for guiding the optimization. To prevent manipulators from acting on multiple objects simultaneously we introduce a capacity constraint of the form

$$C_{\text{cap}} := \sum_{j \in \mathcal{P}} w_{ij}(t) \leq 1\ \forall\, t \in [0, T],$$

which prevents a single manipulator from being fully responsible for multiple objects simultaneously.

E. Constraint manifold

At the core of our formulation are the position and velocity constraints (3)(4). Their most important property becomes apparent only when the constraints are considered over time.

In order to illuminate this property we study the constraints in a one dimensional setting with one free manipulator located at $m : t \to \mathbb{R}$ and a point object located at $p : t \to \mathbb{R}$ and a constant weight w. The position and velocity constraints then reduce to

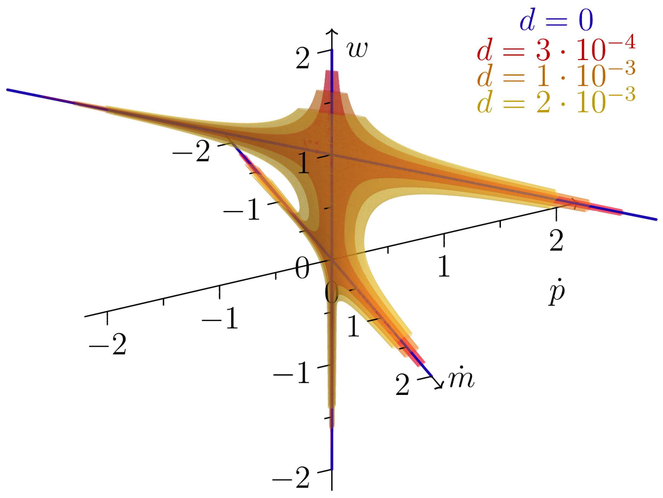


Fig. 3: Level sets of (7). Non-zero velocities of a object-manipulator pair are feasible simultaneously only for $w = 1$.

$$w\,(m(t) - p(t)) = 0$$

and

$$\dot{p}(t) - w\dot{m}(t) = 0 \tag{6}$$

respectively. The constraint violation over time $t \in [t_1, t_2]$ can be quantified using the L^2 norm:

$$d = \int_{t_1}^{t_2} \left(w\,(m(t) - p(t))\right)^2 + \left(\dot{p}(t) - w\dot{m}(t)\right)^2 dt. \tag{7}$$

From (7) we can deduce that:

1) If $w = 0$, the terms corresponding to the position constraint and the manipulator velocity vanish, and thus it must hold that $\dot{p}(t) = 0$.
2) If $w \neq 0$, the position of the manipulator and the object must coincide over the whole time span for the first term to vanish. Since the position of the manipulator and the object must coincide over time, the velocity of the manipulator and the object must be the same. Thus, in order for the velocity terms to cancel out, it must either hold that $w = 1$ or $\dot{p}(t) = \dot{m}(t) = 0$.

This property can be visualized using the level sets of (7) when $m(t_1) = p(t_1)$ (Fig. 3).

Every additional manipulator will add a position constraint as well as a velocity term of the form $-w_i\dot{m}_i(t)$ to (6). Thus, when multiple manipulators are moving the object, all of their velocities must be equal to the velocity of the object, and for the (non-zero) velocity terms to cancel out it must hold that $\sum_i w_i = 1$. Due to the shape of the constraint manifold, the sum of the weights associated with one object will tend towards either 0 or 1. Individual weights between 0 and 1 appear when multiple robots are transporting the same object simultaneously. Our formulation should therefore not be interpreted as a relaxation of the corresponding mixed integer problem where the weights are binary.

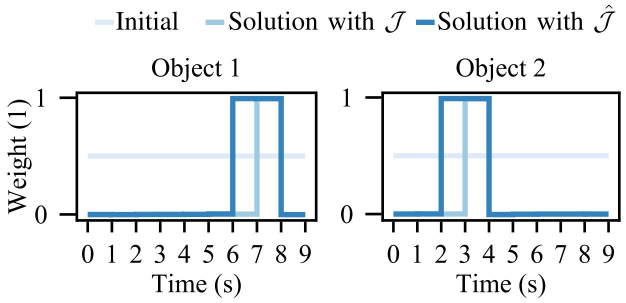


Fig. 4: The association weights for a manipulator moving two objects. The solutions are obtained before and after adding $f_a(\mathbf{p}_1)$ and $f_a(\mathbf{p}_2)$ to the objective function.

F. Nonlinear program

We transform (5) into a nonlinear program in order to solve it numerically. We model the trajectories using cubic Hermite splines on the $N-1$ segments that are defined by N time points. For w_{ij} and r_{dj} we use piece wise constant functions defined on $N-1$ segments. Finally we convert the constrained nonlinear program to an unconstrained problem by applying quadratic and cubic penalty functions to the equality and inequality constraints respectively and solve it using the Gauss-Newton method. We use CHOLMOD [26] for solving the emerging linear system and [25] for collision avoidance. The position and velocity constraints (3)(4) are evaluated at the end points as well as the midpoint of every segment, and the collision constraints at 11 equally spaced points on each segment in order to provide sufficient coverage.

The benefit of the continuous association weights detailed in IV can be demonstrated experimentally. Consider a setup containing two objects that need to be picked up and placed in a different position on the floor, and one manipulator capable of moving the objects. Solving the optimization problem results in a schedule for moving the blocks.

Next we add two additional terms to the objective that captures the acceleration of the objects. We continue the optimization with the updated objective $\hat{\mathcal{J}} := \mathcal{J} + f_a(\mathbf{p}_1) + f_a(\mathbf{p}_2)$. The resulting schedule is shown in Fig. 4 with a trajectory length of 9 segments. As can be seen, the new terms influence the length of the time windows during which the objects are moving. This shows that the emerging schedule is directly affected by the objective function.

V. Results

We evaluate the formulation (5) by applying it to a number of scenarios in a simulated environment. The experiments were implemented in C++ and executed on a desktop computer equipped with an 16-core AMD Ryzen 5950X 3.4 GHz CPU and 32 GB of RAM. All tasks are designed such that they can be solved by the robots in the scene. The association weights w_{ij} are initialized to 0.5 unless stated otherwise.

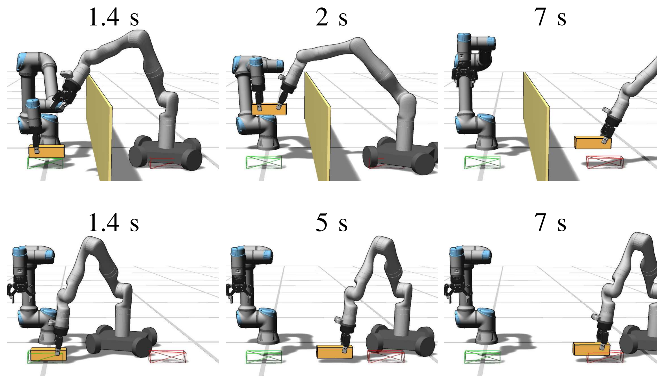


Fig. 5: When the manipulators are separated by a wall the robots must cooperate in order to move the block from the start to the goal pose (outlined in green and red). When the wall is removed the block can be moved from the start to the goal pose by a single robot.

A. Environmental influence

This experiment demonstrates how the allocation of objects to manipulators is affected by obstacles in the environment. The scene consists of one fixed UR5 and a Kinova mounted on an omnidirectional wheeled platform. We use a trajectory consisting of 9 segments, i.e. 10 discrete time points where the joint values of the manipulators and root position of the Kinova platform are initialized to a resting state. The trajectory of the block is initialized such that the final state corresponds to the goal pose while all other states are initialized to the starting pose.

Fig. 5 shows key points of the emerging trajectories. The wall, when present, is placed such that it prevents the Kinova platform from directly reaching for the package and transporting it to the goal position. The robots may thus cooperate, and in the resulting trajectory the block is first lifted by the UR5 and handed over to the Kinova platform, which brings the block to the goal position. When the wall is absent (all other parameters being equal) the task is completed by only the Kinova platform.

B. Multiple moving objects

This experiment consists of three blocks that need to be moved, and four stationary manipulators arranged in a rectangular pattern between the start and goal positions of the blocks. In this experiment we use a trajectory consisting of 13 segments.

The initialization is created by linearly interpolating the trajectories of the blocks and distributing them over time. An illustration of the heuristic schedule is shown in Fig. 6. We then execute ten iterations of the Gauss-Newton solver while keeping the trajectories of the blocks fixed. The resulting manipulator trajectories are then used for initializing the actual optimization where the trajectories of the blocks are included. The solution is captured in Fig. 7. Noteworthy is that the solution contains a segment where one of the blocks is placed to rest on the ground before being picked up again and transported to the goal position. The association weights

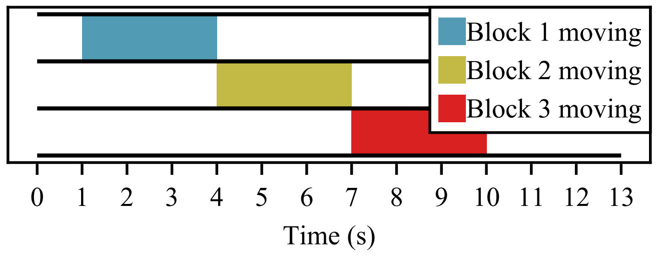


Fig. 6: Tasks involving multiple objects benefit from initialization. We construct the initialization by linearly interpolating the poses of the movable objects within non-overlapping time windows.

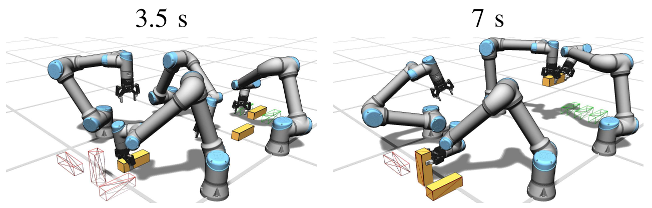


Fig. 7: The second experiment requires multiple handovers for successful completion.

for the blocks are visualized in Fig. 8, showing one block being at rest at time 3-4 s.

C. Including interactive objects

In the previous experiments the starting pose of the blocks are directly reachable by one or more manipulators. However, oftentimes the object of interest can be reached only after manipulating the environment, such as by opening a drawer. Interactive objects can be directly included into (5) as agents subject to appropriate constraints.

We include a drawer by treating it simultaneously as an object that can be moved and as a manipulator equipped with an end effector that can hold objects. The drawer is thus subject to a velocity constraint which allows it to move only when actuated by another manipulator, and a corresponding pose constraint attached to the handle of the drawer (see (4) and (3)). The start pose of the block is inside of the closed drawer while the goal pose is on the table next to the robot as shown in Fig. 9. We initialize the trajectory of the block by linearly interpolating between the start and the goal poses. The trajectory of the drawer is initialized to the resting pose, i.e. closed, while the trajectory of the UR5 is initialized to

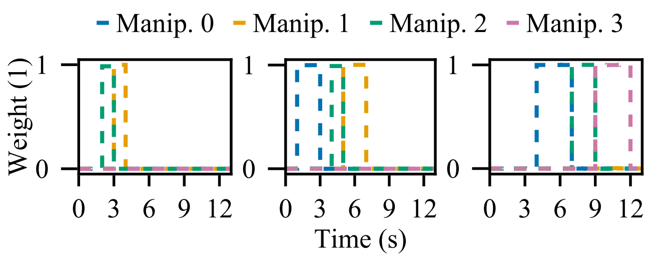


Fig. 8: Association weights between the manipulators and the blocks as a function of time. At time 3-4 s the middle block is placed on the ground before being picked up again.

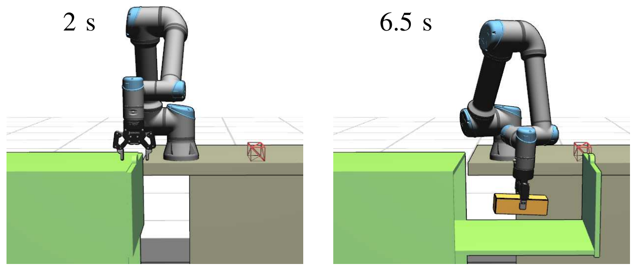


Fig. 9: The formulation also supports interactive objects. Here the drawer needs to be opened before the block can be retrieved.

the rest state. The association weight between the drawer and the block is initialized to 1. The resulting trajectory features the UR5 opening the drawer before reaching for the block and placing it on the table.

D. Extension for multiple grasping orientations

The the formulation in IV provides some flexibility in choosing the grasp pose through the longitudinal offset δ_{ij} and the angle θ_{ij}, but the grasping orientation is still fixed around the two remaining axes of the object. However, the blocks used in the previous examples allow four distinct grasping orientations around the longitudinal axis. In this section we show how the formulation in IV can be extended in order to enable all of these orientations.

We introduce the functions $\gamma_{ijk} : [0, T] \rightarrow [0, 1]$ that denote the association of a particular grasping orientation k between an object j and a manipulator i. The position constraints (3) can now be replaced with

$$\gamma_{ijk}(t) w_{ij}(t) \left(\mathbf{f}_i(\mathbf{m}_i) - \hat{\mathbf{g}}_k(\mathbf{p}_j, \mathbf{y}_{ij})\right) = \mathbf{0},$$

where $\hat{\mathbf{g}}_k$ is the target grasp pose after applying the offset corresponding to index k. By also setting $\sum_k \gamma_{ijk}(t) = 1 \; \forall \; i \in \mathcal{M}$ we can ensure that at least one alternative will be active.

The updated formulation can be used to solve e.g. reorientation problems. In Fig. 10, two manipulators are tasked to reorient a block such that the face that is initially facing upwards will be facing towards the ground at the end. The values of γ_{ijk} are shown in Fig. 11. The reorientation of the block could in this case be completed with only one handover, however, the obtained solution is a local minimum featuring four handovers. In this experiment the trajectory of the block has been initialized to a linearly interpolated trajectory between the start and the goal pose while the manipulator trajectories are initialized to the rest pose.

E. Weight derivative limit

As discussed in IV-E, the formulation enables multiple robots to seamlessly transport a single package. We can induce this behavior by introducing an additional constraint that limits the rate of change of the association weights w_{ij}. This constraint is of the form

$$C_{\Delta_w^+} := \dot{w}_{ij} \leq \Delta_w^+$$
$$C_{\Delta_w^-} := \dot{w}_{ij} \geq \Delta_w^-$$

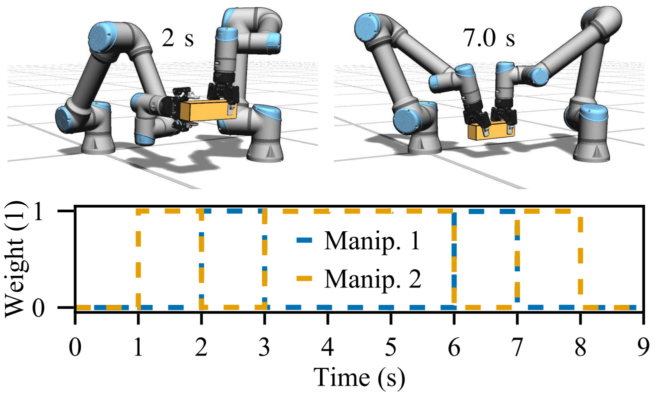


Fig. 10: Here two robots are reorienting a block such that the face that is initially facing upwards will be facing towards the ground.

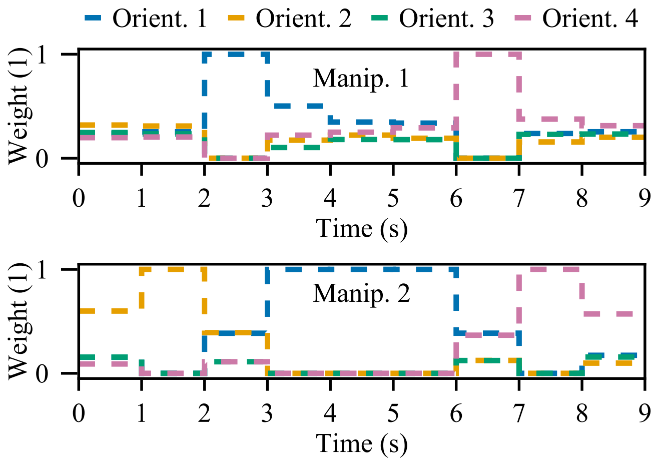


Fig. 11: The orientation weights $\gamma_{ijk}(t)$ as a function of time. Manipulator 1 is grasping the block from two different directions while manipulator 2 is using three different ones. The constraint becomes active when both $\gamma_{ijk}(t)$ and $w_{ij}(t)$ are greater than zero.

where Δ_w^+ and Δ_w^- denote the upper and lower limit of the derivatives.

This constraint causes the pick-up and release phases to be extended. The weights now need $\frac{1}{\Delta_w^+}$ time units to switch from 0 to 1, which can be useful e.g. in order to provide enough time for the grippers to open and close during a pick-up or a handover. As discussed in IV-E, the velocity can be non-zero only when the weights of one object sum up to one, and therefore the velocity of both the end effector and the object will be zero when the object is picked up and dropped on the ground. During a handover the object may still move as long as the weights sum up to 1. An experiment with two UR5s where the upper and lower derivative limits are set to $\frac{1}{2}$ and $-\frac{1}{2}$ respectively is shown in Fig. 12.

F. Optimization runtime

Finally we study how increasing the number of manipulators impacts the optimization. In this experiment we assemble UR5s on a line with the task of moving a block

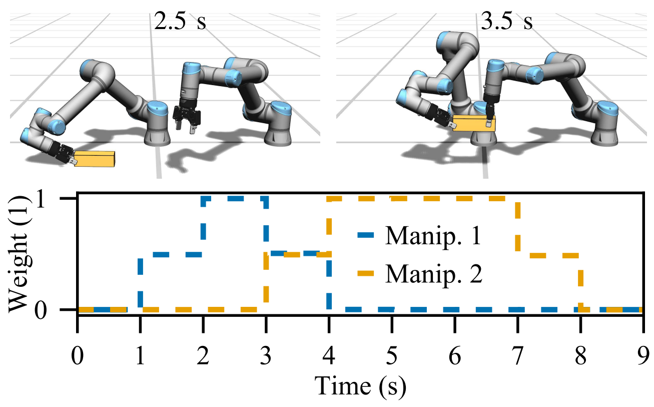
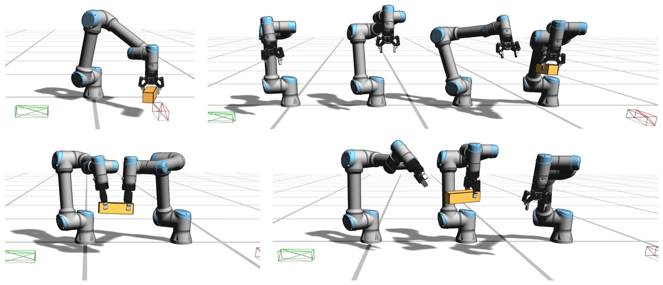


Fig. 12: Two UR5s performing a handover where the switching time is constrained. During time segments 1-2 s and 7-8 s, the sum of the association weights is less than 1, and as discussed in IV-E, the velocity constraint (4) then ensures that the object stays in place.



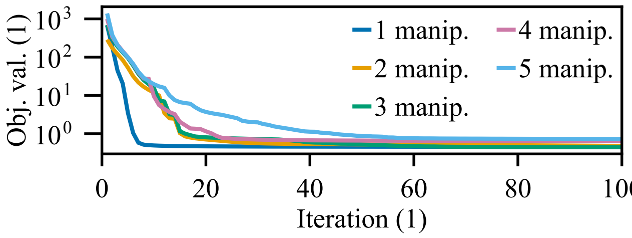


Fig. 13: The objective value as a function of the iteration number with different numbers of manipulators.

from the start of the line to the end. Adding manipulators increases the number of iterations needed for convergence, as shown in Fig. 13. The five manipulator setup is shown in Fig. 1. The runtime and the number of variables for each experiment is shown in I. The table must be interpreted carefully as the exact task description has a significant impact on the optimization landscape and thus also the number of iterations needed for convergence. The measurements reported in I also include rendering and are intended as rough estimates only.

VI. Discussion and Conclusion

The results in V show that decisions regarding high level actions such as pick, place and open can be obtained implicitly by solving a nonlinear optimization problem. The actions emerge automatically as part of the solution without the need to manually specify which manipulator should be working, or even when.

TABLE I: Number of variables and average runtime for different number of manipulators. The reported runtime is the average of four measurements.

Manipulators	1	2	3	4	5
Wall time (s)	9.2	20.3	28.5	36.5	46.3
Number of variables	305	436	567	698	829

Trajectory optimization involving collision avoidance is in general a non-convex problem. Gradient based methods use only local information and may therefore struggle in finding new trajectories that lie far away from the initial guess. This can be demonstrated by constructing a variant of the drawer experiment from V-C where the goal position of the block is placed on top of the drawer which did not converge to any reasonable solution in our experiments. This problem has also been identified in [20]. Finding the global optimum of the problem requires more sophisticated optimization algorithms that are capable of exploring the optimization space efficiently. Existing TAMP algorithms may in some cases be able to work around this problem by including the actions themselves into the problem formulation, turning the actions into conditions for solving the optimization problem.

Additionally it is not entirely clear how the formulation could be extended to support objects with inherently discrete states, e.g. light switches, while still being continuous. Our formulation also does not have any notion of temporal precedence, i.e. objects may arrive at the goal position in an arbitrary order. This can, however, be mitigated to some extent by careful initialization.

We believe that the formulation presented here can be useful, especially as part of a larger TAMP algorithm. Even when combined with existing TAMP algorithms, by finding some actions implicitly it would be possible to reduce the number of actions that must be considered during the sequencing. As the optimization method used in this work might have difficulties in highly non-convex problems we would additionally like to investigate the use of sampling based methods for handling the end effector constraints.

References

[1] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 3684–3691.

[2] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning." in *IJCAI*, 2015, pp. 1930–1936.

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

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

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

[6] W. Thomason and R. A. Knepper, "A unified sampling-based approach to integrated task and motion planning," in *The International Symposium of Robotics Research*. Springer, 2019, pp. 773–788.

[7] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Sampling-based methods for factored task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1796–1825, 2018.

[8] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, pp. 265–293, 2021.

[9] F. Lagriffoul, N. T. Dantam, C. Garrett, A. Akbari, S. Srivastava, and L. E. Kavraki, "Platform-independent benchmarks for task and motion planning," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3765–3772, 2018.

[10] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, eaat8414, 2019.

[11] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, "A survey on learning-based robotic grasping," *Current Robotics Reports*, vol. 1, pp. 239–249, 2020.

[12] M. Suomalainen, Y. Karayiannidis, and V. Kyrki, "A survey of robot manipulation in contact," *Robotics and Autonomous Systems*, vol. 156, 104224, 2022.

[13] R. Shome and K. E. Bekris, "Anytime multi-arm task and motion planning for pick-and-place of individual objects via handoffs," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2019, pp. 37–43.

[14] H. I. Bozma and M. Kalalıoğlu, "Multirobot coordination in pick-and-place tasks on a moving conveyor," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 4, pp. 530–538, 2012.

[15] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (ToG)*, vol. 31, no. 4, pp. 1–8, 2012.

[16] I. Mordatch, Z. Popović, and E. Todorov, "Contact-invariant optimization for hand manipulation," in *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, 2012, pp. 137–144.

[17] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[18] K. Hauser and J.-C. Latombe, "Multi-modal motion planning in non-expansive spaces," *The International Journal of Robotics Research*, vol. 29, no. 7, pp. 897–915, 2010.

[19] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[20] R. Takano, H. Oyama, and M. Yamakita, "Continuous optimization-based task and motion planning with signal temporal logic specifications for sequential manipulation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8409–8415.

[21] Y. V. Pant, H. Abbas, and R. Mangharam, "Smooth operator: Control using the smooth robustness of temporal logic," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2017, pp. 1235–1240.

[22] M. P. Bendsøe, "Optimal shape design as a material distribution problem," *Structural optimization*, vol. 1, pp. 193–202, 1989.

[23] G. I. Rozvany, "Aims, scope, methods, history and unified terminology of computer-aided topology optimization in structural mechanics," *Structural and Multidisciplinary optimization*, vol. 21, pp. 90–108, 2001.

[24] S. Zimmermann, G. Hakimifard, M. Zamora, R. Poranne, and S. Coros, "A multi-level optimization framework for simultaneous grasping and motion planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2966–2972, 2020.

[25] S. Zimmermann, M. Busenhart, S. Huber, R. Poranne, and S. Coros, "Differentiable collision avoidance using collision primitives," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 8086–8093.

[26] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, "Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate," *ACM Transactions on Mathematical Software (TOMS)*, vol. 35, no. 3, pp. 1–14, 2008.