Handling implicit and explicit constraints in manipulation planning

Joseph Mirabel, Florent Lamiraux

Handling implicit and explicit constraints in manipulation planning

Joseph Mirabel
LAAS
University of Toulouse, CNRS
Toulouse, France
Email: joseph.mirabel@laas.fr

Florent Lamiraux
LAAS
University of Toulouse, CNRS
Toulouse, France
Email: florent.lamiraux@laas.fr

***Abstract*—This paper deals with manipulation planning. The problem consists in automatically computing paths for a system composed of one or several robots, with one or several grippers and one or several objects that can be grasped and moved by the robots. The problem gives rise to constraints that can be either explicit – an object is in a gripper – or implicit – an object is hold by two different grippers. This paper proposes an algorithm that handles such sets of constraints and solves them in an explicit way as much as possible. When all constraints cannot be made explicit, substitution is operated between variables to make the resulting implicit constraint with as few variables as possible. The manipulation planning problem is modelled as a constraint graph that stores all the constraints of the problem.**

I. Introduction

Manipulation planning is an instance of path planning where objects are moved by robot grippers. This implies a lot of constraints on the motion of the system composed of the robots and of the objects. Namely, when an object is not grasped, it should remain still in a stable placement, while when an object is grasped by a gripper, it should move in a rigid manner with the gripper. Those constraints can be expressed numerically on the configuration variables of the robot and object. As such, constraint resolution is a key feature one must take care of [17]. It has a direct influence on the completeness [3] and the efficiency [13] of the overall approach.

Pioneering work in motion planning [31, 1] have been initiated in the late 1980's. Random path planning methods developed in the 1990's [15] have been first adapted to the manipulation planning problem by Siméon et al. [26]. Recently, manipulation planning has regained interest with various approaches that tackle the inherent complexity of the problem. The domain is traditionally divided into several categories.

In navigation among movable obstacles (NAMO), the robot needs to move obstacles in order to reach the goal configuration [29, 22, 5]. Rearrangement planning consists in automatically finding a sequence of manipulation paths that move several objects from initial configurations to specified goal configurations [18, 23, 19]. Other research papers investigate multi-arm motion planning [9, 10, 6]. From a geometric perspective, manipulation planning is a hybrid problem the configuration space of which is a union of subspaces – called state – defined by constraints (gripper A holds object B for instance) on the positions of objects and robots. Motion within or from one to another state are also subject to numerical constraints. Mirabel et al. [20] propose a representation of the problem using a graph the nodes of which are states and the edges of which are manipulation paths. Nodes and edges both store numerical constraints.

Similar graph structures are also more or less explicitly used in various papers [3, 14, 11]. The partially discrete nature of the problem has also given rise to approaches that integrate task and motion planning techniques [4, 2, 7, 27, 12, 8]. Toussaint and Lopes [30] combine branch and bound with non-linear optimization techniques.

A manipulation path as computed in [20] is a concatenation of elementary linear interpolations subject to numerical constraints. To evaluate a configuration along such a path, the linear interpolation is first computed and then, the resulting configuration is projected on the sub-manifold that satisfies the constraints related to the current manipulation edge (Figure 1). This projection is performed numerically by iteratively solving the tangent linear system (Newton-Raphson). During path planning, a lot of constrained path are produced and evaluated for collision checking. It is therefore important that constraint resolution takes as little time as possible.

The contribution of this paper is a method that solves a set of numerical constraints. The method derives from [3, 5]. Some of the constraints are explicit, some are implicit. Resolution is performed via substitution of variables that can be explicitly expressed with respect to other variables, thus defining a non-linear constraint with less variables. The main benefit is a gain in terms of time of computation since explicit constraints are much faster to solve.

The paper is organized as follows. In Section II, we define implicit and explicit constraints, we explain how implicit constraints are numerically solved and we exhibit sufficient conditions under which the combination of two explicit constraints is still an explicit constraint. In Section III, we describe our algorithm that simplifies the resolution of a set of constraints by substituting variables that can be computed with respect to others and then by applying Newton-Raphson method. Section IV is devoted to the extension of our work to functions with values in the space of rigid transformations $SE(3)$. Finally, section V provides some experimental results showing

the benefit of using explicit representation of constraints when possible.

II. Problem definition

Let us consider a manipulation planning problem with R robots and N objects. The configuration space of the whole system is the Cartesian product of the configuration spaces of the robots and of the objects:

$$\mathcal{C} = \mathcal{C}_{r_1} \times \cdots \times \mathcal{C}_{r_R} \times \mathcal{C}_{o_1} \times \cdots \times \mathcal{C}_{o_N}.$$

For simple objects, $\mathcal{C}_{o_i}$ is the space of rigid-body motions denoted by $SE(3)$, but more complex objects like a desk with drawers for instance, are represented by a kinematic chain.

A solution to a manipulation planning problem is a continuous curve in $\mathcal{C}$, starting from the initial configuration of the system and ending in a configuration satisfying the goal specifications (position of the robot, position of the objects, or both). Unlike for the classical path planning problem, any curve in the configuration space is almost never a manipulation planning path since some constraints have to be satisfied along the manipulation path. See [16] for an overview of the various instances of path planning problems with constraints.

A. Numerical constraint

More precisely, let us denote by $(\mathbf{q}_{r_1}, \cdots, \mathbf{q}_{r_R}, \mathbf{q}_{o_1}, \cdots, \mathbf{q}_{o_N}) \in \mathcal{C}$, the configuration of the whole system. When robot $i \in \{1, \cdots, R\}$ is grasping object $j \in \{1, \cdots, N\}$, the position o_j completely depends on the position of robot i, and the positions of all other objects are fixed:

$$\mathbf{q}_{o_j} = f_1(\mathbf{q}_{r_i}) \tag{1}$$
$$\mathbf{q}_{o_k} = \mathbf{p}_k \in \mathcal{C}_{o_k} \quad \forall k \in \{1, \cdots, N\}, k \neq j \tag{2}$$

where f_1 is a smooth mapping from $\mathcal{C}_{r_i}$ to $\mathcal{C}_{o_j}$, and $\mathbf{p}_k$ are constant positions of the objects that are not manipulated.

The above constraint is said to be *explicit* since some configuration variables (of the objects) can be computed with respect to others (of the robot).

Definition 1. *An* implicit constraint *is defined by a smooth mapping g from $\mathcal{C}$ to a vector space $\mathbb{R}^m$. A configuration $\mathbf{q} \in \mathcal{C}$ is said to satisfy constraint g for threshold ε iff*

$$\|g(\mathbf{q})\| \leq \varepsilon.$$

Definition 2. Combination of implicit constraints. *Let m_1, m_2 be two positive integers, let g_1 from $\mathcal{C}$ to $\mathbb{R}^{m_1}$ and g_2 from $\mathcal{C}$ to $\mathbb{R}^{m_2}$ be two smooth mappings. The combination of the implicit constraints defined by g_1 and g_2 is the implicit constraint defined by the mapping g from $\mathcal{C}$ to $\mathbb{R}^{m_1+m_2}$ that maps to $\mathbf{q} \in \mathcal{C}$ the vector*

$$\begin{pmatrix} g_1(\mathbf{q}) \\ g_2(\mathbf{q}) \end{pmatrix}.$$

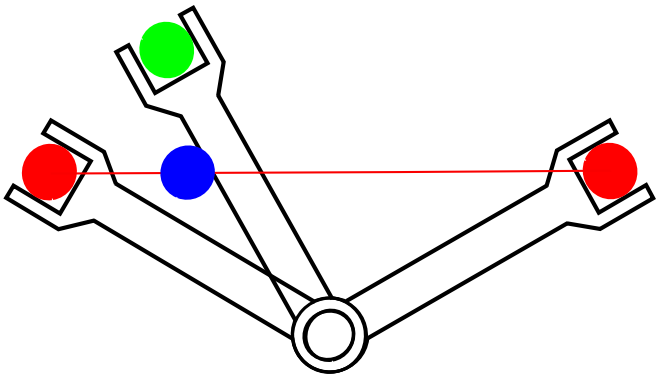


Fig. 1. Manipulation path: simple example of a 1-dof robot grasping a disc. The path between two configurations is defined by linear interpolation of the 1-dof robot and of the ball position (in blue). Then the constraint (position of the disc in the gripper) is solved – explicitly if possible or by a Newton-Raphson like algorithm – (in green).

B. Implicit constraint resolution

As mentioned in the introduction, in [20], manipulation paths are concatenations of linear interpolations projected on the sub-manifold defined by manipulation constraints. In other words, evaluating a configuration along such a path consists in first computing the linear interpolation and then to solve the manipulation constraints from this configuration to obtain a configuration that satisfies the constraints. An important consequence is that paths are not discretized but implicitly defined everywhere on their definition interval. Figure 1 shows a simple example.

Algorithm 1 `implicit-constraint-solver`($\mathbf{q}$, g, ε): find $\mathbf{q}$ such that $\|g(\mathbf{q})\| < \varepsilon$

$\alpha = .1$, $i = 0$, $\alpha_{max} = .95$
while $\|g(\mathbf{q})\| > \varepsilon$ and $i \leq 20$ **do**
  // $(.)^+$ denotes the Moore-Penrose pseudo-inverse
  $\mathbf{q} \leftarrow \mathbf{q} - \alpha \left(\frac{\partial g}{\partial \mathbf{q}}(\mathbf{q})\right)^+ g(\mathbf{q})$
  $i \leftarrow i + 1$
  // Make α tend toward α_{max}
  $\alpha \rightarrow \alpha_{max} - .8 * (\alpha_{max} - \alpha)$
end while
if $\|g(\mathbf{q})\| \leq \varepsilon$ **then**
  return $\mathbf{q}$
else
  return failure
end if

Definition 3. *An Implicit Constraint Solver is a mapping that takes as input an implicit constraint g, a tolerance threshold ε and that returns a mapping h from a part D of $\mathcal{C}$ to $\mathcal{C}$ such that*

$$\forall \mathbf{q} \in D, \|g(h(\mathbf{q}))\| < \varepsilon$$

h is called a *projector on the sub-manifold satisfying g*. The procedure described by Algorithm 1 previously introduced

in [3, 5] is an implicit constraint solver. In this case, D is the subspace of configuration for which the procedure converges to a solution. Note that the pseudo-inverse is computed by a singular value decomposition of complexity $O(m^2n + mn^2 + n^3)$ where n is the dimension of the configuration space and m is the dimension of non-linear constraints. Constraint substitution as described in Section III makes both the dimension of constraints and the number of variables decrease in the Newton-Raphson resolution.

C. Explicit constraints

We denote by

- $\dim \mathcal{C}$ the dimension of the configuration vectors in $\mathcal{C}$,
- $I_{\mathcal{C}}$ the set of positive integers not greater than $\dim \mathcal{C}$,
- I a subset of $I_{\mathcal{C}}$,
- $\bar{I}$ the complement in $I_{\mathcal{C}}$ of I,
- $|I|$ the cardinal of I.

If $\mathbf{q} \in \mathcal{C}$ is a configuration, we denote by $\mathbf{q}_I \in \mathbb{R}^{|I|}$ the vector composed of the components of $\mathbf{q}$ of increasing indices in I.

a) Example: if $\mathbf{q} = (q_1, q_2, q_3, q_4, q_5, q_6)$ and $I = \{1, 3, 6\}$, then $\mathbf{q}_I = (q_1, q_3, q_6)$, $\mathbf{q}_{\bar{I}} = (q_2, q_4, q_5)$.

Definition 4. *An explicit constraint $E = (in, out, f)$ is a mapping from $\mathcal{C}$ to $\mathcal{C}$, defined by the following elements:*

- *a subset of input indices $in \subset \{1, \cdots, \dim \mathcal{C}\}$,*
- *a subset of output indices $out \subset \{1, \cdots, \dim \mathcal{C}\}$,*
- *a smooth mapping f from $\mathbb{R}^{|in|}$ to $\mathbb{R}^{|out|}$,*

satisfying the following properties:

- $in \cap out = \emptyset$,
- *for any $\mathbf{p} \in \mathcal{C}$, $\mathbf{q} = E(\mathbf{p})$ is defined by*

$$\mathbf{q}_{\bar{out}} = \mathbf{p}_{\bar{out}}$$
$$\mathbf{q}_{out} = f(\mathbf{p}_{in}).$$

With the same notation as above, if we define g as the mapping from $\mathcal{C}$ to $\mathbb{R}^{|out|}$ defined by

$$g(\mathbf{p}) = \mathbf{p}_{out} - f(\mathbf{p}_{in}), \tag{3}$$

E is a projector on the submanifold satisfying g with threshold equal to 0.

In other words, an explicit constraint can be formulated as an implicit constraint.

D. Combination of explicit constraints

We now establish some sufficient conditions under which the combination of 2 explicit constraints yields an explicit constraint and we show how to compute the resulting explicit constraint.

Let $E_1 = (in_1, out_1, f_1)$ and $E_2 = (in_2, out_2, f_2)$ be 2 distinct explicit constraints. If

$$out_1 \cap out_2 = \emptyset \text{ and } (in_1 \cap out_2 = \emptyset \text{ or } in_2 \cap out_1 = \emptyset), \tag{4}$$

the combination of both explicit constraints yields another explicit constraint $E = (f, in, out)$ with input and output sets in and out defined later on.

Without loss of generality, by switching the constraints E_1 and E_2, we can reduce hypothesis (4) to

$$out_1 \cap out_2 = \emptyset \quad \text{and} \quad in_1 \cap out_2 = \emptyset. \tag{5}$$

Under this assumption, the input and output spaces of f are defined by

$$in = (in_1 \cup in_2) \setminus (in_2 \cap out_1),$$
$$out = out_1 \cup out_2.$$

To evaluate E, we build a tuple T of $\dim \mathcal{C}$ elements initialized to $\emptyset$. For each

- $i \in out_1$, we set T [i] to E_1,
- $i \in out_2$, we set T [i] to E_2,

where $T[i]$ denotes the i-th element of T. Figures 2 gives an example.

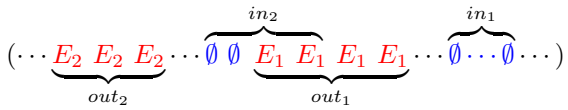


Fig. 2. Combination E of two compatible explicit constraints E_1 and E_2. In red, output variables of E; in blue, input variables of E.

To compute $\mathbf{q} = E(\mathbf{p})$, for $\mathbf{p} \in \mathcal{C}$, we first copy $\mathbf{p}$ into $\mathbf{q}$, we evaluate $f_1(\mathbf{q}_{in_1})$ and update $\mathbf{q}_{out_1}$ with the result, and then we evaluate $f_2(\mathbf{q}_{in_2})$ and update $\mathbf{q}_{out_2}$ with the result.

III. Combination of implicit and explicit constraints

Let k_g and k_E be two non-negative integers. Given a set of k_g implicit constraints $g_1, \cdots, g_{k_g}$ and k_E explicit constraints $E_1, \cdots E_{k_E}$, we wish to build a projector on the submanifold that satisfies all these constraints. We proceed as follows.

Let E be the first explicit constraint, and let g be the combination of $g_1, \cdots, g_{k_g}$. For each j from 2 to k_E.

- if E and E_j satisfy hypothesis (4), we replace E by the combination of E and E_j,
- otherwise, we replace g by the combination of g with the implicit representation (3) of E_j.

After this operation, we have one implicit constraint g with values in $\mathbb{R}^m$, and at most one explicit constraint $(in, out, f) \triangleq E$ that represent the same solution submanifold as $g_1, \cdots, g_{k_g}$ and $E_1, \cdots E_{k_E}$ together.

$$g(\mathbf{q}) = 0 \tag{6}$$
$$\mathbf{q}_{out} = f(\mathbf{q}_{in}) \tag{7}$$

A. Constraint resolution

Up to variable reordering, we assume that output variables and input variables are contiguous:

$$\mathbf{q} = (\mathbf{q}_{out}, \mathbf{q}_{in}, \mathbf{q}_{\overline{in \cup out}})$$

System (6-7) becomes

$$g(f(\mathbf{q}_{in}), \mathbf{q}_{in}, \mathbf{q}_{\overline{in \cup out}}) = 0 \tag{8}$$
$$\mathbf{q}_{out} = f(\mathbf{q}_{in}) \tag{9}$$

We define $\tilde{g}$ the mapping from $\mathbb{R}^{|\bar{out}|}$ to $\mathbb{R}^m$ as

$$\tilde{g}(\mathbf{q}_{in}, \mathbf{q}_{\overline{in \cup out}}) = g(f(\mathbf{q}_{in}), \mathbf{q}_{in}, \mathbf{q}_{\overline{in \cup out}})$$

To solve (8), we apply Algorithm 1 to $\tilde{g}$. Note that $\tilde{g}$ has less variables than g. We need to compute the Jacobian of $\tilde{g}$ with respect to $\mathbf{q}_{\bar{out}} = (\mathbf{q}_{in}, \mathbf{q}_{\overline{in \cup out}})$:

$$\begin{aligned}\frac{\partial \tilde{g}}{\partial \mathbf{q}_{in}}(\mathbf{q}_{\bar{out}}) &= \frac{\partial g}{\partial \mathbf{q}_{out}}(f(\mathbf{q}_{in}), \mathbf{q}_{\bar{out}}) \frac{\partial f}{\partial \mathbf{q}_{in}}(\mathbf{q}_{in}) \\ &\quad + \frac{\partial g}{\partial \mathbf{q}_{in}}(f(\mathbf{q}_{in}), \mathbf{q}_{\bar{out}}) \\ \frac{\partial \tilde{g}}{\partial \mathbf{q}_{\overline{in \cup out}}}(\mathbf{q}_{\bar{out}}) &= \frac{\partial g}{\partial \mathbf{q}_{\overline{in \cup out}}}(f(\mathbf{q}_{in}), \mathbf{q}_{\bar{out}})\end{aligned}$$

As g is a combination of implicit constraint, its Jacobian is simply obtained by stacking the Jacobians of all the implicit constraints. When f is obtained from the combination of several explicit constraints, computing $\partial f / \partial \mathbf{q}_{in}$ is not as trivial.

B. Jacobian of the combination of explicit constraints

	a	b	c
E_2			
E_2	$\frac{\partial f_2}{\partial \mathbf{q}_d} \frac{\partial f_{1\,d}}{\partial \mathbf{q}_a}$	$\frac{\partial f_2}{\partial \mathbf{q}_b}$	$\frac{\partial f_2}{\partial \mathbf{q}_c}$
E_2			
E_1			
E_1	$\frac{\partial f_1}{\partial \mathbf{q}_a}$	$\frac{\partial f_1}{\partial \mathbf{q}_b}$	0
E_1			
E_1			

Fig. 3. Jacobian of the combination of two explicit constraints. $f_{1\,d}$ denotes the components of f_1 corresponding to elements of $d = in_2 \cap out_1$.

As in Section II-D, we denote by E the combination of two explicit constraints E_1 and E_2. The rows of the Jacobian of f correspond to output variables $\mathbf{q}_{out_1}$ and $\mathbf{q}_{out_2}$. The columns correspond to input variables with indices in $(in_1 \cup in_2) \backslash (in_2 \cap out_1)$. We define the following disjoint subsets of indices:

$$\begin{aligned}a &= in_1 \setminus in_2, \\ b &= in_1 \cap in_2, \\ c &= in_2 \setminus in_1 \setminus out_1, \\ d &= in_2 \cap out_1.\end{aligned}$$

With these definitions, we have

$$\begin{aligned}in_1 &= a \cup b \\ in_2 &= b \cup c \cup d \\ in &= a \cup b \cup c\end{aligned}$$

Note that in Figure 2, $d = in_1 \cap in_2 = \emptyset$.

To compute the Jacobian of f, we separate the variables:

$$f(\mathbf{q}_{in}) = \begin{pmatrix} f_1(\mathbf{q}_a, \mathbf{q}_b, \mathbf{q}_c) \\ f_2(\mathbf{q}_a, \mathbf{q}_b, \mathbf{q}_c) \end{pmatrix}$$

To make notation tractable, we make an assumption on the order of the output variables and the input variables. The implementation is of course more general as explained in the next section. Note that in figures 3 and 2, the order of output variables is different.

f_1 only depends on $\mathbf{q}_a$ and $\mathbf{q}_b$. The Jacobian rows corresponding to f_1 thus contain the Jacobian of f_1. f_2 depends on $\mathbf{q}_b$, $\mathbf{q}_c$, and $\mathbf{q}_d$. The blocks corresponding to f_2 output and $\mathbf{q}_b$, $\mathbf{q}_c$ input contain the Jacobian of f_2, while variables $\mathbf{q}_d$ are output variables of f_1 the corresponding block is thus a product of Jacobians. Figure 3 provide expressions of the blocks that compose the Jacobian of f.

C. Implementation details

To compute the value and Jacobian of a combination of explicit constraints, we have implemented a class called `MatrixBlockView`. The constructor takes as input two sets of indices `rows` and `cols`. For i, $0 \leq i < |\texttt{rows}|$ and j, $0 \leq j < |\texttt{rows}|$, let I be the $i-1$-th element of `rows` and J be the $j-1$-th element of `cols`. If

- `M = MatrixXd` is a matrix of the Eigen library, then
- `Mv = MatrixBlockView (M, rows, cols)` emulates the Eigen matrix such that
- `Mv (i,j)` is a reference to `M (I,J)` element.

The Jacobian matrix `J` of explicit constraint f in the previous section represented in Figure 3 is thus computed as follows. `J_1` and `J_2` are the Jacobian matrices of f_1 and f_2.

```
MatrixBlockView(J, out_2, a) = MatrixBlockView(J_2, all, d)
                               * MatrixBlockView(J1, d, a)
MatrixBlockView(J, out_2, b) = MatrixBlockView(J_2, all, b)
MatrixBlockView(J, out_2, c) = MatrixBlockView(J_2, all, c)
MatrixBlockView(J, out_1, a) = MatrixBlockView(J_1, all, a)
MatrixBlockView(J, out_1, b) = MatrixBlockView(J_1, all, b)
MatrixBlockView(J, out_1, c) = 0
```

IV. Function valued in SE(3)

For clarity, the developments of the previous sections make the assumption that functions take values in $\mathbb{R}^m$ where m is a positive integer. In equation (1) however, function f_1 represents the position of a rigid object and takes value in $SE(3)$[1]. Although the position of a rigid object can be represented by 6 values $(x, y, z, roll, pitch, yaw)$, it is well known that this representation suffers a lot of shortcomings.

To represent an element of $SE(3)$, we use a 7-dimensional vector $(q_0, \cdots q_6)$ such that:

- (q_0, q_1, q_2) represents the translation, and
- $\|(q_3, q_4, q_5, q_6)\| = 1$, quaternion $q_3 i + q_4 j + q_5 k + q_6$ encodes the rotation.

The velocity of a rigid object, called a twist can be represented by a 6-dimensional vector $(\mathbf{v}, \omega)$ where

- $\mathbf{v} \in \mathbb{R}^3$ is the velocity of the point of the object that coincides with the origin of the world frame, expressed in the basis of the object,
- $\omega \in \mathbb{R}^3$ is the angular velocity of the object expressed in the basis of the object.

See [21] chapter 2, Section 3.2 for details.

[1]The group of rigid-body transformations

A. Addition and subtraction

Let $(\mathbf{v}, \omega) \in \mathbb{R}^6$ represent a twist and $\mathbf{q} = (q_0, \cdots q_6)$ represent an element of $SE(3)$. Applying the constant twist $(\mathbf{v}, \omega)$ during unit time to $\mathbf{q}$ leads to another element of $SE(3)$ that we denote by:

$$\mathbf{p} = \mathbf{q} + (\mathbf{v}, \omega)$$

This definition of the addition is an extension of the addition on vector spaces where applying a constant velocity $\mathbf{v}$ during unit time starting from a vector $\mathbf{q}$ leads to $\mathbf{q} + \mathbf{v}$.

We define the subtraction accordingly: if $\mathbf{q}_1$ and $\mathbf{q}_2$ are two elements of $SE(3)$, $\mathbf{q}_2 - \mathbf{q}_1$ is a vector of $\mathbb{R}^6$ representing the (minimal) constant twist leading from $\mathbf{q}_1$ to $\mathbf{q}_2$ in unit time.

With this definition, we can extend the developments of the previous sections to functions with values in Cartesian products of vector spaces and $SE(3)$.

V. Results

Our implementation of the above method is integrated in the package `hpp-constraints` of the open-source platform HPP [20]. All tests are run on a processor Intel Xeon 3.5 Ghz, with 8 Go of RAM.

To analyse the influence of explicit constraint substitution into Newton-Raphson algorithm (Alg. 1), we compare our approach to the standard Newton-Raphson algorithm, as it is the most successful method [28, 3, 5] for general non-linear constraints.

A. Construction set

In this scenario, two UR3 robots, $(R_i)_{i\in\{0,1\}}$ assemble a set of magnetised cylinders $(C_i)_{i\in\{0,1\}}$ and spheres $(S_i)_{i\in\{0,1\}}$ (Figure 4). The configuration space of the system is composed of vectors of the following form:
$\mathbf{q} = (\mathbf{q}_{R_0}, \mathbf{q}_{R_1}, \mathbf{x}_{C_0}, \mathbf{p}_{C_0}, \mathbf{x}_{C_1}, \mathbf{p}_{C_1}, \mathbf{x}_{S_0}, \mathbf{p}_{S_0}, \mathbf{x}_{S_1}, \mathbf{p}_{S_1})$, where

- $\mathbf{q}_{R_0}, \mathbf{q}_{R_1} \in \mathbb{R}^6$ represent the configurations of each robot,
- $\mathbf{x}_{C_0}, \mathbf{x}_{C_1}, \mathbf{x}_{S_0}, \mathbf{x}_{S_1} \in \mathbb{R}^3$ represent the positions of the centers of the cylinders and spheres,
- $\mathbf{p}_{C_0}, \mathbf{p}_{C_1}, \mathbf{p}_{S_0}, \mathbf{p}_{S_1} \in \mathbb{R}^4$ are unit quaternions representing the orientations of the cylinders and spheres.

We denote

$$I_{R_0} = \{0, \cdots, 5\},\ I_{R_1} = \{6, \cdots, 11\}, I_{C_0} = \{12, \cdots, 18\},$$
$$I_{C_1} = \{19, \cdots, 25\}, I_{S_0} = \{26, \cdots, 32\}, I_{S_1} = \{33, \cdots, 39\}$$

the sets of indices corresponding to the configurations of respectively the robots, the cylinders, and the spheres.

The goal is to produce a sequence of manipulation paths in order to assemble two spheres on a cylinder.

We provide to the algorithm the high-level task sequence as follows:

1) R_0 grasps S_0,
2) R_1 grasps C_0,
3) S_0 is assembled on C_0,
4) R_0 releases S_0,
5) R_0 grasp S_1,
6) S_1 is assembled on C_0,
7) R_0 releases S_1,
8) R_1 releases C_0.

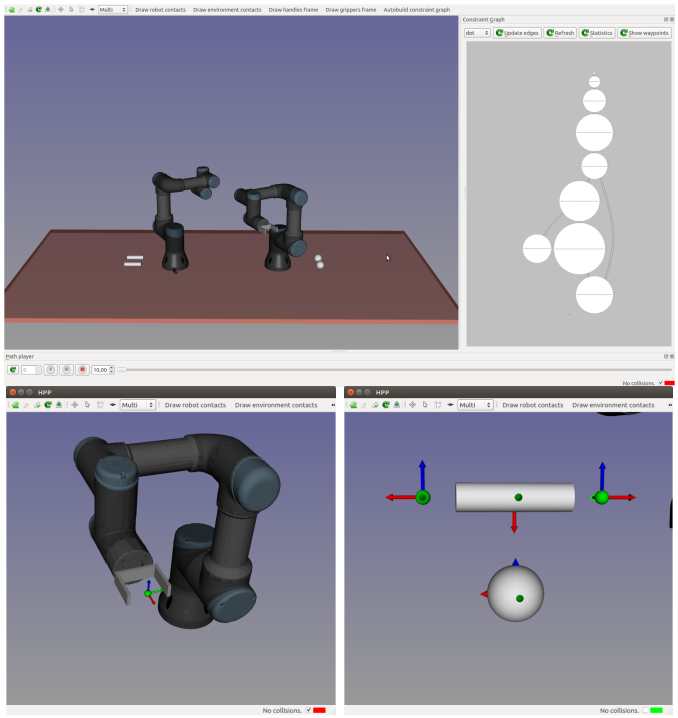


Fig. 4. Top left: two UR-3 robots assemble two spheres on a cylinder. The complete resolution takes a few seconds. Top right: the constraint graph is composed of 9 states. Bottom: grasp constraints are defined by frames attached to robot grippers and to objects. The middle frame of the cylinder defines the grasp constraint by the robot gripper. The left and right frames correspond to positions of the spheres when stuck to the cylinder.

Given this sequence of actions, our software builds a constraint graph the nodes of which are states of the system and the edges of which are transitions that encode path constraints. Each state encodes a set of position constraints between robots and objects. The states are the following

1) all objects lie on the ground,
2) $R_0 \to S_0$,
3) $R_0 \to S_0$ and $R_1 \to C_0$,
4) $R_0 \to S_0$ and $R_1 \to C_0$ and $S_0 \leftrightarrow C_0$,
5) $R_1 \to C_0$ and $S_0 \leftrightarrow C_0$,
6) $R_0 \to S_1$ and $R_1 \to C_0$ and $S_0 \leftrightarrow C_0$,
7) $R_0 \to S_1$ and $R_1 \to C_0$ and $S_0 \leftrightarrow C_0$ and $S_1 \leftrightarrow C_0$,
8) $R_1 \to C_0$ and $S_0 \leftrightarrow C_0$ and $S_1 \leftrightarrow C_0$,
9) $S_0 \leftrightarrow C_0$ and $S_1 \leftrightarrow C_0$ and assembly lies on the ground.

where $\to$ means "grasps", and $\leftrightarrow$ means "is stuck on". Any configuration of any manipulation path lies in one of the above states. Each of states 1), 2), 3), 5), 6), 8), and 9) is composed of explicit constraints that are all compatible. For instance, in state 8), the position of C_0 depends on the configuration of R_1, and the positions of S_0, S_1 depend on the position of C_0. Hence, the configuration of R_0 uniquely determines the positions of C_0, S_0, and S_1. On the contrary, states 4) and 7) form closed kinematic chains and require an implicit resolution. Our work handles all those sets of constraints in a

seamless and efficient way.

a) Variable substitution: let us describe the variable substitution algorithm for state 4). We denote by E_1, E_2, E_3 the constraints in this state:

$$E_1 = R_0 \to S_0 \quad (\mathbf{x}_{S_0}, \mathbf{p}_{S_0}) = f_{R_0 \to S_0}(\mathbf{q}_{R_0})$$
$$E_2 = R_1 \to C_0 \quad (\mathbf{x}_{C_0}, \mathbf{p}_{C_0}) = f_{R_1 \to C_0}(\mathbf{q}_{R_1})$$
$$E_3 = S_0 \leftrightarrow C_0 \quad (\mathbf{x}_{S_0}, \mathbf{p}_{S_0}) = f_{S_0 \leftrightarrow C_0}(\mathbf{x}_{C_0}, \mathbf{p}_{C_0})$$

We first compose E_1 and E_2. Using the notation of Section II-D, we have

$$in_1 = I_{R_0},\ out_1 = I_{S_0},\ in_2 = I_{R_1},\ out_2 = I_{C_0}.$$

We notice immediatly that Condition (4) is satisfied and therefore E_1 and E_2 are compatible. They combine into another explicit constraint $E = (in, out, f)$ such that

$$out = I_{S_0} \cup I_{C_0},\ in = I_{R_0} \cup I_{R_1}$$
$$f(\mathbf{q}_{R_0}, \mathbf{q}_{R_1}) = \begin{pmatrix} f_{R_0 \to S_0}(\mathbf{q}_{R_0}) \\ f_{R_1 \to C_0}(\mathbf{q}_{R_1}) \end{pmatrix}$$

Using the same reasoning, we can state that E and E_3 are not compatible since (Condition 4)

$$out \cap out_3 = (I_{S_0} \cup I_{C_0}) \cap I_{S_0} \neq \emptyset.$$

We therefore apply the constraint resolution algorithm described in Section III. After substitution, constraints (6-7) rewrite

$$f_{R_0 \to S_0}(\mathbf{q}_{R_0}) - f_{S_0 \leftrightarrow C_0}(f_{R_1 \to C_0}(\mathbf{q}_{R_1})) = 0$$
$$(\mathbf{x}_{S_0}, \mathbf{p}_{S_0}) = f_{R_0 \to S_0}(\mathbf{q}_{R_0})$$
$$(\mathbf{x}_{C_0}, \mathbf{p}_{C_0}) = f_{R_1 \to C_0}(\mathbf{q}_{R_1})$$

Implicit resolution is performed on the first line of the above system using algorithm (1). Note that this constraint of dimension 6 implies only 12 degrees of freedom $(\mathbf{q}_{R_0}, \mathbf{x}_{C_0}, \mathbf{p}_{C_0})$ out of 36.[2]

To compute a solution path, we generate target configurations in each node, accessible from the initial configuration, and then we connect those targets by a visibility-PRM algorithm [25] in the appropriate sub-manifold. This is a variant of the method described in [24]. The manipulation planning algorithm is not the topic of this paper.

We analyse the improvements of the projection algorithm. From a set of 10000 random configurations, we compute the average computation time and success ratio of the projection with and without explicit constraint substitution. Figure 5 shows the results. As expected, explicit constraint substitution increases the success ratio and reduces the computation time. Note that in states 4) and 7), although constraints are solved implicitly, we observe an improvement due to the reduction of the number of variables. The accompanying video shows a solution path found to assemble the parts.

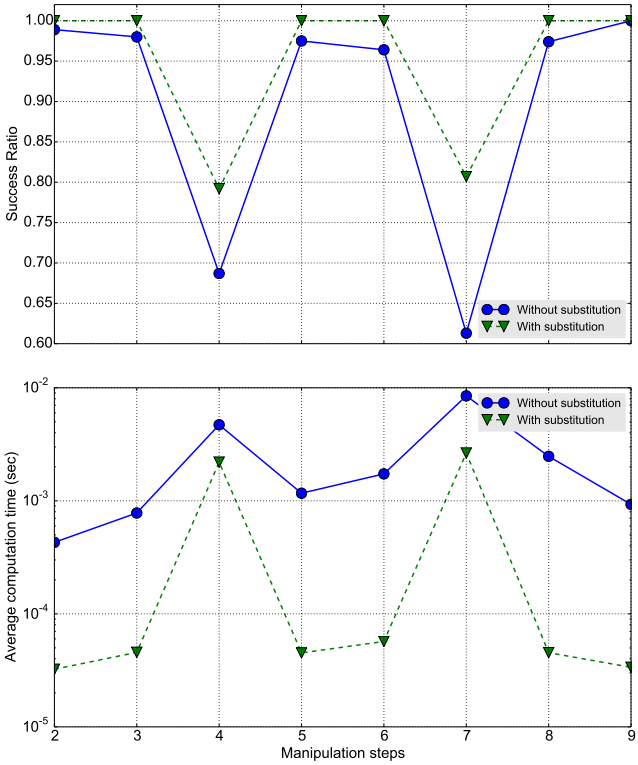
B. Romeo humanoid robot

In this scenario, the humanoid robot Romeo[3] manipulates a placard with both hands, $Hand_{right}$ and $Hand_{left}$. Two

[2]Poses in SE(3) stand for 6 degrees of freedom although the are represented by 7 numbers.

[3]The model can be obtained at http://doc.aldebaran.com/2-5/family/romeo/index_romeo.html

Fig. 5. Projection success ratio and computation time for the construction set problem. We compare fully implicit formulation (without substitution) to our combination of implicit and explicit constraint resolution (with substitution). Note the logarithmic scale.



frames, $Placard_{high}$ and $Placard_{low}$, are defined on the placard, as shown on Figure 6. The three following states are considered:

1) $Hand_{right}$ grasps $Placard_{high}$,
2) $Hand_{right}$ grasps $Placard_{high}$ and $Hand_{left}$ grasps $Placard_{low}$,
3) $Hand_{left}$ grasps $Placard_{low}$,

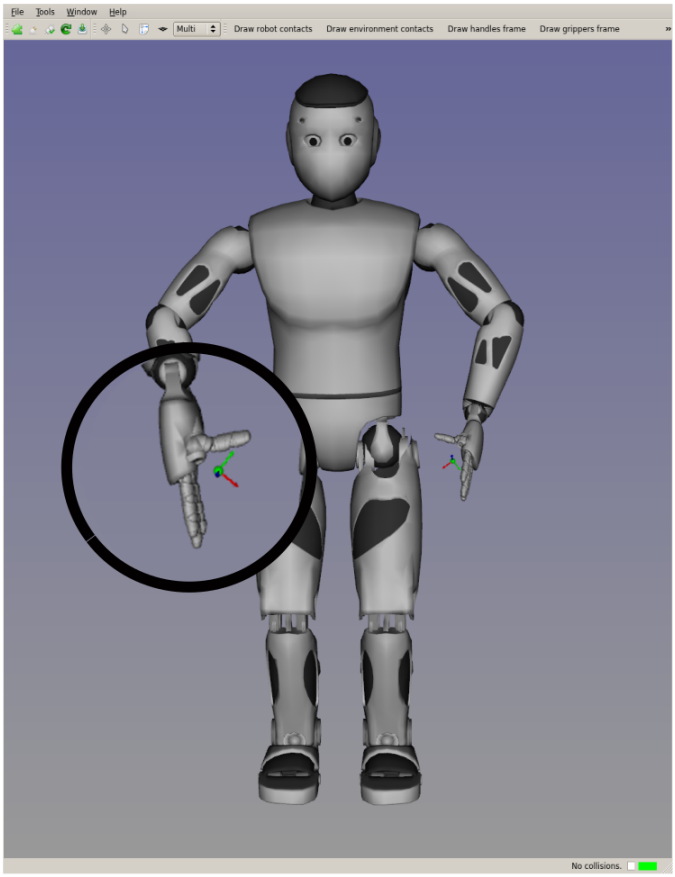
Moreover, in all of these states, the following additional constraints are added to ensure robot balance:

- feet at fixed positions (dimention 12),
- robot center of mass above support polygon (dimension 2).

To our best knowledge, there is no simple explicit formulation of these constraints so a solver based on exact inverse kinematics would not be able to address this problem.

a) Variable substitution: let us describe the variable substitution algorithm for state 2) above. The configuration vector is of the form $\mathbf{q} = (\mathbf{q}_{romeo}, \mathbf{q}_{placard})$ where $\mathbf{q}_{romeo} \in \mathbb{R}^{40}$, and $\mathbf{q}_{placard} \in \mathbb{R}^7$. The constraints are of the following form:

$$\mathbf{q}_{placard} = f_{left}(\mathbf{q}_{romeo})$$
$$\mathbf{q}_{placard} = f_{right}(\mathbf{q}_{romeo})$$
$$f_{balance}(\mathbf{q}_{romeo}, \mathbf{q}_{placard}) = 0$$



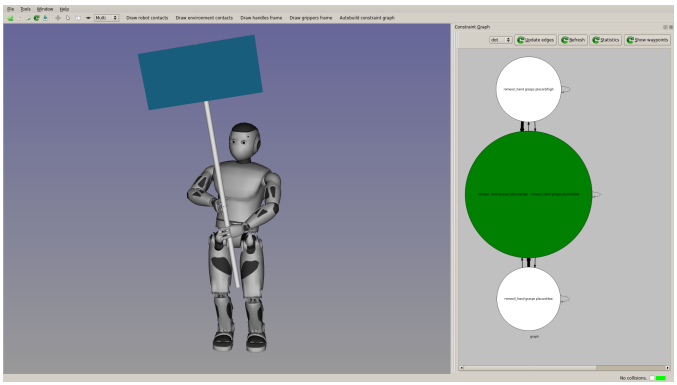


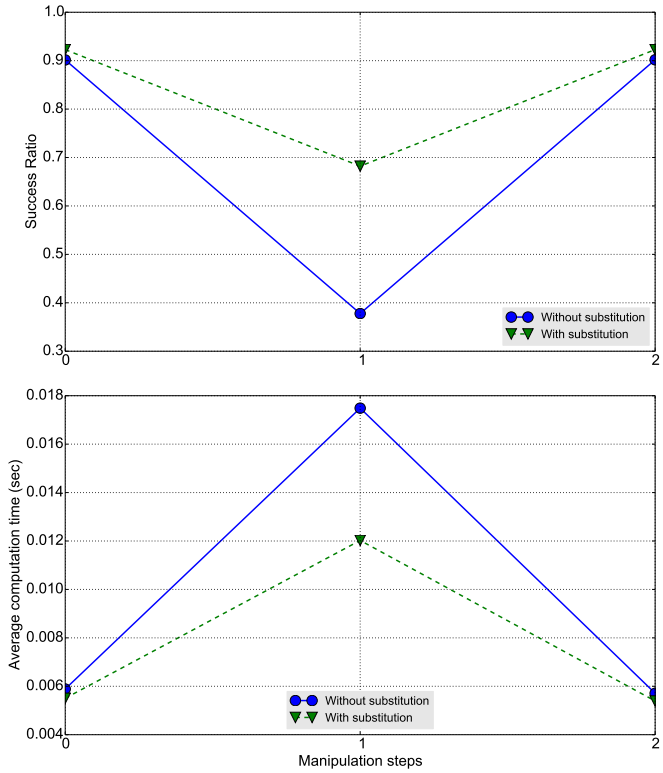
Fig. 6. Top: Romeo robot with the frames attached to each hand. Bottom left: Romeo holds the placard with boths hands. Bottom left: the three states considered are shown on the constraint graph, on the right.

Note that $f_{balance}$ output is of dimension 14. The first two constraints are explicit but incompatible. The system thus becomes

$$\begin{aligned} &\mathbf{q}_{placard} = f_{left}(\mathbf{q}_{romeo}) \\ &f_{left}(\mathbf{q}_{romeo}) - f_{right}(\mathbf{q}_{romeo}) = 0 \\ &f_{balance}(\mathbf{q}_{romeo}, f_{left}(\mathbf{q}_{romeo})) = 0 \end{aligned}$$

The last two constraints are solved implicitely. They represent an implicit constraint of dimension 20 with 40 variables, instead of dimension 26 with 46 variables.

Fig. 7. Projection success ratio and computation time for humanoid robot problem. We compare fully implicit formulation (without substitution) to our combination of implicit and explicit constraint resolution (with substitution).



From a set of 10000 random configurations, we compare the success ratio and the average computation time of the projection algorithm with and without explicit constraint substitution. Figure 7 shows the results. For states 0) and 2), the success ratio and the computation time are both marginally influenced. In these states, the placard is held by one hand. For state 1), substitution has a great positive influence on the success ratio and computation time of the projection.

VI. Conclusion

This paper proposes a method to solve automatically non-linear constraints expressed either implicitly or explicitly. Such constraints are in the core of the manipulation planning problem, especially when manipulation paths go through states where an object is grasped by several grippers.

Our method is integrated with success in a manipulation planning framework and experimental results show that huge improvement can be obtained.

It can of course be argued that the example provided in Section V-A can be solved with exact inverse kinematics. However, we claim that our approach applies to more general cases where exact inverse kinematics is not available, like in Section V-B where a humanoid robot in quasi-static equilibrium grasps objects.

ACKNOWLEDGMENTS

This work has been partially funded by the FLAG-ERA JTC project ROBOCOM++, which aims at rethinking robotics for the robot companion of the future. It has received funding from the European Community Seventh Framework Programme (FP7/2007-2013) under grant agreement 608849.

REFERENCES

[1] Rachid Alami, Thierry Siméon, and Jean-Paul Laumond. A geometrical approach to planning manipulation tasks. the case of discrete placements and grasps. In *5th International Symposium on Robotics Research*, Tokyo, Japan, 1989.

[2] Jennifer Barry, Leslie Kaelbling, and Tomás Lozano-Pérez. A hierarchical approach to manipulation with diverse actions. In *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013.

[3] Dmitry Berenson, Siddhartha S Srinivasa, and James Kuffner. Task space regions: A framework for pose-constrained manipulation planning. *The International Journal of Robotics Research*, page 0278364910396389, 2011.

[4] Stéphane Cambon, Rachid Alami, and Fabien Gravot. A hybrid approach to intricate motion, manipulation and task planning. *International Journal of Robotics Research*, 28, 2009.

[5] S. Dalibard, A. El Khoury, F. Lamiraux, A. Nakhaei, M. Taïx, and J.-P. Laumond. Dynamic walking and whole-body motion planning for humanoid robots: an integrated approach. *The International Journal of Robotics Research*, 32(9-10):1089–1103, 2013.

[6] Andrew Dobson and Kostas Bekris. Planning representations and algorithms for prehensile multi-arm manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015.

[7] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Ffrob: An efficient heuristic for task and motion planning. In *Workshop on the Algorithmic Foundations of Robotics*, 2014.

[8] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Sampling-based methods for factored task and motion planning. *arXiv preprint arXiv:1801.00680*, 2018.

[9] Mamoun Gharbi, Juan Cortés, , and Thierry Siméon. Roadmap composition for multi-arm systems path planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Saint-Louis, USA, 2009.

[10] K. Harada, T. Tsuji, and J.-P. Laumond. A manipulation motion planner for dual-arm industrial manipulators. in proceedings of. In *IEEE International Conference on Robotics and Automation*, pages 928–934, Hongkong, China, 2014.

[11] Kris Hauser and Victor Ng-Thow-Hing. Randomized multi-modal motion planning for a humanoid robot manipulation task. *The International Journal of Robotics Research*, 30(6):678–698, 2011.

[12] Giray Havur, Guchan Ozbilgin, Esra Erdem, and Volkan Patoglu. Hybrid reasoning for geometric rearrangement of multiple movable objects on cluttered surfaces. In *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 2014.

[13] Léonard Jaillet and Josep M Porta. Path planning under kinematic constraints by rapidly exploring manifolds. *IEEE Transactions on Robotics*, 29(1):105–117, 2013.

[14] Sören Jentzsch, Andre Gaschler, Oussama Khatib, and Alois Knoll. MOPL: A multi-modal path planner for generic manipulation tasks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, September 2015. http://youtu.be/1QRvjBw58bU.

[15] Lydia E. Kavraki, P. Svestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, August 1996.

[16] Zachary Kingston, Mark Moll, and Lydia Kavraki. Sampling-based methods for motion planning with constraints. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[17] Zachary K. Kingston, Mark Moll, and Lydia E. Kavraki. Sampling-based methods for motion planning with constraints. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018. Accepted for publication.

[18] Athanasios Krontiris and Kostas Bekris. Dealing with difficult instances of object rearrangement. In *Robotics Science and Systems*, Roma, Italy, 2015.

[19] Puttichai Lertkultanon and Quang-Cuong Pham. A single-query manipulation planner. *IEEE Robotics and Automation Letters*, 1(1):198–205, 2015.

[20] Joseph Mirabel, Steve Tonneau, Pierre Fernbach, Anna-Kaarina Seppälä, Mylène Campana, Nicolas Mansard, and Florent Lamiraux. HPP: a new software for constrained motion planning. In *IEEE/RSJ Intelligent Robots and Systems*, October 2016.

[21] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[22] Dennis Nieuwenhuisen, A Frank van der Stappen, and Mark H Overmars. An effective framework for path planning amidst movable obstacles. In *Algorithmic Foundation of Robotics VII*, pages 87–102. Springer, 2008.

[23] Jun Ota. Rearrangement of multiple movable objects-integration of global and local planning methodology. In *2004 IEEE International Conference on Robotics and Automation*, volume 2, pages 1962–1967. IEEE, 2004.

[24] Philipp Sebastian Schmitt, Werner Neubauer, Wendelin Feiten, Kai M. Wurm, Georg v. Wichert, and Wolfram Burgard. Optimal, sampling-based manipulation planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[25] T. Simeon, J.-P. Laumond, and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. *Journal of Advanced Robotics*, 14(6):477–494, 2000.

[26] Thierry Siméon, Jean-Paul Laumond, Juan Cortés, and Anis Sahbani. Manipulation planning with probabilistic roadmaps. *International Journal of Robotics Research*, 23(7/8), July 2004.

[27] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[28] Mike Stilman. Global manipulation planning in robot joint space with task constraints. *IEEE Transactions on Robotics*, 26(3):576–584, 2010.

[29] Mike Stilman and James Kuffner. Planning among movable obstacles with artificial constraints. *The International Journal of Robotics Research*, 27(11-12):1295–1307, 2008.

[30] Marc Toussaint and Manuel Lopes. Multi-bound tree search for logic-geometric programming in cooperative manipulation domains. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[31] Gordon Wilfong. Motion planning in the presence of movable obstacles. In *Proceedings of the fourth annual symposium on Computational geometry*, pages 279–288. ACM, 1988.