Sample-Based Methods for Factored Task and Motion Planning

Caelan Reed Garrett
MIT CSAIL
Cambridge, MA 02139
caelan@csail.mit.edu

Tomás Lozano-Pérez
MIT CSAIL
Cambridge, MA 02139
tlp@csail.mit.edu

Leslie Pack Kaelbling
MIT CSAIL
Cambridge, MA 02139
lpk@csail.mit.edu

***Abstract*—There has been a great deal of progress in developing probabilistically complete methods that move beyond motion planning to multi-modal problems including various forms of task planning. This paper presents a general-purpose formulation of a large class of discrete-time planning problems, with hybrid state and action spaces. The formulation characterizes conditions on the submanifolds in which solutions lie, leading to a characterization of robust feasibility that incorporates dimensionality-reducing constraints. It then connects those conditions to corresponding conditional samplers that are provided as part of a domain specification. We present domain-independent sample-based planning algorithms and show that they are both probabilistically complete and computationally efficient on a set of challenging benchmark problems.**

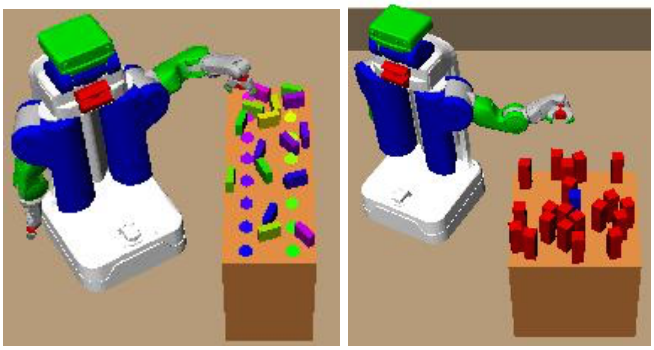

Fig. 1. Left: experiment 1. Right: experiment 3.

I. Introduction

Many important robotic domains of interest require planning in a very high-dimensional space that includes not just the robot configuration, but also the "configuration" of the external world state, including a variety of quantities such as object poses, reaction states of chemical or biological processes, or intentions of other agents. There has been a great deal of progress in developing probabilistically complete sampling-based methods that move beyond motion planning to multi-modal problems including various forms of task planning. These new methods each require a new formulation, definition of robust feasibility, sampling methods, and search algorithm. This paper presents a general-purpose formulation of a large class of discrete-time planning problems, with continuous or hybrid state and action spaces.

The primary theoretical contribution of this paper is a formulation of factored transition systems that exposes the topology of their solution space, particularly in the presence of dimensionality-reducing constraints. The key insight is that, in some cases, the intersection of solution constraint manifolds is itself a manifold that can be identified using only the individual constraint manifolds. By understanding the topology of the solution space, we can define a robust feasibility property that characterizes a large class of problems for which sampling-based planning methods can be successful.

The primary algorithmic contribution is the construction of two sample-based planning algorithms that exploit the factored, compositional structure of the solution space to draw samples from a space in which solutions have positive measure. These algorithms search in a combined space that includes the discrete structure (which high-level operations, such as "pick" or "place" happen in which order) and parameters (particular continuous parameters of the actions) of a solution. Theoretically, these algorithms are probabilistically complete when given sufficient samplers. Practically, they can solve complex instances of task-and-motion planning problems.

II. Related Work

Planning problems in which the goal is not just to move the robot without collision but also to operate on the objects in the world have been addressed from the earliest days of motion planning to this day, for example [28, 27, 40, 2, 1, 33, 35, 36, 38, 17, 3, 23, 15, 10]. In recent years, there have been a number of approaches to integrating discrete task planning and continuous motion planning [5, 32, 21, 12, 13, 31, 7, 37], aimed at increasing the capabilities of autonomous robots.

Hauser et al. [18] introduced a framework and algorithm for probabilistically complete multi-modal motion planning. Vega-Brown et al. [39] extended these ideas to optimal planning with differential constraints.

Lagriffoul et al. [25, 26] interleave the search for a discrete action sequence and the geometric parameters and focus on limiting the amount of geometric backtracking. They generate a set of approximate linear constraints imposed by a plan skeleton under consideration, *e.g.*, from grasp and placement choices, and use linear programming to compute a valid assignment or determine that one does not exist. Lozano-Pérez and Kaelbling [29] take a similar approach but leverage constraint satisfaction problem (CSP) solvers to identify

satisfactory geometric parameters from discretized domains.

Srivastava et al. [34] integrate task and motion planning by designing an interface allowing an off-the-shelf motion planner to share geometric information with an off-the-shelf task planner. Their approach first plans at the task-level and then attempts to produce motion plans satisfying the discrete actions. If an induced motion planning problem is infeasible, the task-level planning repeats with new action preconditions identifying the source of the infeasibility.

The FFRob algorithm of Garrett et al. [14, 16] is related to the INCREMENTAL algorithm discussed in this paper; it also involves sampling a fixed set of object poses and robot configurations and then planning with them. An iterative version of FFRob is probabilistically complete and exponentially convergent [16]. However, the approach in FFRob is specialized to a particular class of pick-and-place problems and does not use the planning to guide the sampling.

Dantam et al. [6] formulate task and motion planning as a satisfiability modulo theories (SMT) problem. They use an incremental constraint solver to add motion constraints to the task-level logical formula when a candidate task plan is found. Upon failure, they iteratively increase the plan depth and motion planning timeouts resulting in a probabilistically complete algorithm for fixed placements and grasps.

III. Factored Transition System

We begin by defining a general class of models for controllable discrete-time continuous-space dynamical systems. It is possible to address many continuous-time problems in this framework, as long as they can be solved with a finite sequence of continuous control inputs. A discrete-time *transition system* $\mathcal{S} = \langle \mathcal{X}, \mathcal{U}, \mathcal{T} \rangle$ is defined by a set of states (*state-space*) $\mathcal{X}$, set of controls (*control-space*) $\mathcal{U}$, and a *transition relation* $\mathcal{T} \subseteq \mathcal{X} \times \mathcal{U} \times \mathcal{X}$. A *problem* $\mathcal{P} = \langle x^0, X^*, \mathcal{S} \rangle$ is an initial state $x^0 \subseteq \mathcal{X}$, a set of goal states $X^* \subseteq \mathcal{X}$, and a transition system $\mathcal{S}$. A *plan* for a problem $\mathcal{P}$ is finite sequence of k control inputs $(u^1, ..., u^k)$ and k states $(x^1, ..., x^k)$ such that $(x^{i-1}, u^i, x^i) \in \mathcal{T}$ for $i \in \{1, ..., k\}$ and $x^k \in X^*$.

A. Factoring

We consider *factored transition systems* with state-spaces $\bar{\mathcal{X}} = \mathcal{X}_1 \times ... \times \mathcal{X}_m$ and control-spaces $\bar{\mathcal{U}} = \mathcal{U}_1 \times ... \times \mathcal{U}_n$ that are defined by m *state variables* $\bar{x} = (x_1, ..., x_m)$ and n *control variables* $\bar{u} = (u_1, ..., u_n)$. The transition relation is a subset of the *transition parameter space* $\mathcal{T} \subseteq \bar{\mathcal{X}} \times \bar{\mathcal{U}} \times \bar{\mathcal{X}}$ Valid transitions are $(x_1, ..., x_m, u_1, ..., u_n, x'_1, ..., x'_m) \in \mathcal{T}$. To simplify notation, we will generically refer to each x_v, u_v, or x'_v in a transition as a *parameter* z_p where $p \in \Theta = \{1, ..., 2m + n\}$ indexes the entire sequence of variables. For a subset of parameter indices $P = (p_1, ..., p_k)$, define $\bar{z}_P = (z_{p_1}, ..., z_{p_k}) \in \bar{\mathcal{Z}}_P$ to be the combined values and $\bar{\mathcal{Z}}_P = \mathcal{Z}_{p_1} \times ... \times \mathcal{Z}_{p_k}$ to be the combined domain of the parameters.

Many transition relations are hybrid, in that there is a discrete choice between different types of operation, each of which has a different continuous constraint on the relevant parameters. For example, a pick-and-place problem has transitions corresponding to a robot moving its base, picking each object, and placing each object. In order to expose the discrete structure, we decompose the transition relation $\mathcal{T} = \bigcup_{a=1}^{\alpha} T_a$ into the union of α smaller transition relations T_a.

A transition relation T_a often is the intersection of several *constraints* on a subset of the transition parameters. A constraint is a pair $C = \langle P, R \rangle$ where $P \subseteq \Theta$ is a subset of parameters and $R \subseteq \bar{\mathcal{Z}}_P$ is a relation on these parameters. For instance, *pick* transitions involve constraints that the end-effector initially is empty, the target object is placed stably, the robot's configuration forms a kinematic solution with the placement, and the end-effector ultimately is holding the object. A constraint decomposition is particularly useful when $|P| << |\Theta|$; *i.e.*, each individual constraint has low arity. Let $\widehat{C} = \{\bar{z} \in \bar{\mathcal{Z}} \mid \bar{z}_P \in R\}$ be the extended form of the constraint.

We represent each T_a as a *conjunctive clause* of β constraints $\mathcal{C}_a = \{C_1, ..., C_\beta\}$ where $T_a = \bigcap_{C \in \mathcal{C}_a} \widehat{C}$. Within a clause, there are implicit domain constraints on each parameter z_p of the form $C = \langle (p), \mathcal{Z}_p \rangle$. The transition relation $\mathcal{T}$ is the union of α conjunctive constraint clauses $\{\mathcal{C}_1, ..., \mathcal{C}_\alpha\}$.

Factoring the transition relation can expose constraints that have a simple equality form. Equality constraints are important because they transparently reduce the dimensionality of the transition parameter space. A *constant equality* constraint $C = \langle (p), \{\kappa\} \rangle$ (denoted $z_p = \kappa$) indicates that parameter p has value κ. A *pairwise equality* constraint $C = \langle (p, p'), \{(z, z) \mid z \in \mathcal{Z}_p \cap \mathcal{Z}_{p'}\} \rangle$ (denoted $z_p = z_{p'}$) indicates that parameters p, p' have the same value.

For many high-dimensional systems, the transition relation is *sparse*, meaning its transitions only alter a small number of the state variables at a time. Sparse transition systems have transition relations where each conjunctive clause contains pairwise equality constraints $x_v = x'_v$ for most variables v. Intuitively, most actions do not change many state variables.

The initial state $\bar{x}^0$ and set of goal states $\bar{X}^*$ can be specified using conjunctive clauses $\mathcal{C}_0$ and $\mathcal{C}_*$ defined solely on state variables. Because $\bar{x}^0$ is a single state, its clause is composed of just constant equality constraints.

B. Constraint Satisfaction

Planning can be thought of as a combined search over discrete clauses and hybrid parameter values. For each action there is a choice of a discrete operation type and of continuous parameters. To select a type is to select a clause from the transition relation; to select its parameters is to select the x, u, x' values. A finite sequence of clauses $\vec{a} = (a_1, ..., a_k)$ is a *plan skeleton* [29]. For example, solutions to pick-and-place problems are sequences of *move*, *pick*, *move-while-holding*, and *place* clauses involving the same object.

The *plan parameter space* for a plan skeleton $\vec{a} = (a_1, ..., a_k)$ is an alternating sequence of states and actions $(\bar{x}^0, \bar{u}^1, \bar{x}^1, ..., \bar{u}^k, \bar{x}^k) = \bar{z} \in \bar{\mathcal{X}} \times (\bar{\mathcal{U}} \times \bar{\mathcal{X}})^k = \bar{\mathcal{Z}}$. Again, we generically refer to each variable in the plan parameter space as z_p where now $p \in \Theta = \{1, ..., m + k(m + n)\}$. When applying the constraints for clause a_t, plan parameter x^{t-1} is

transition parameter x and likewise x^t is x'. Solutions using this skeleton must satisfy a single conjunctive clause of all plan-wide constraints $\mathcal{C}_{\vec{a}} = \mathcal{C}_0 \cap \mathcal{C}_{a_1} \cap ... \cap \mathcal{C}_{a_k} \cap \mathcal{C}_*$.

Given a plan skeleton, finding a set of valid parameter values is a classical *constraint satisfaction problem*. The joint set of constraints forms a *constraint network*, a bipartite graph between constraints and parameters [8, 26]. An edge between a constraint node $C_b^{a_t}$ and state or control node x_v^{t-1}, u_v^t, or x_v^t is defined if and only if $v \in P_b^{a_t}$. Figure 2 displays a general constraint network. Many transition systems in practice will have constraint networks with many fewer edges because each $P_b^{a_t}$ contains only a small number of parameters.

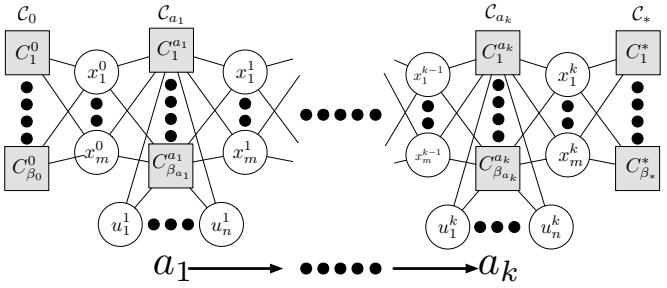

Fig. 2. A constraint network for a generic plan skeleton $\vec{a} = (a_1, ..., a_k)$.

IV. Example Domains

We are interested in a general algorithmic framework that can be applied in many *domains*. A domain $\mathcal{D} = \{\mathcal{P}, ...\}$ is loosely defined as a set of problems that share the same constraint forms and variable forms. Consider the following two domains and their representation as factored transition systems. We begin with a motion planning example to illustrate the approach, and then describe a pick-and-place problem.

A. Motion Planning

Many motion planning problems may be defined by a bounded configuration space $\mathcal{Q} \subset \mathbb{R}^d$ and collision-free configuration space $Q_{free} \subseteq \mathcal{Q}$. A motion between configurations q and q' is valid if the straight-line trajectory between them is collision free: $CFree = \{(q, q') \in \mathcal{Q}^2 \mid \forall t \in [0, 1].\ tx + (1 - t)x' \in Q_{free}\}$. Problems are given by an initial configuration $q^0 \in \mathcal{Q}$ and a goal configuration $q^* \in \mathcal{Q}$.

Motion planning can be modeled as a transition system with state-space $\bar{\mathcal{X}} = \mathcal{Q}$ and action-space $\bar{\mathcal{U}} = \emptyset$. The transition relation has a single clause $\{\mathcal{C}_{Step}\}$. Clause $\mathcal{C}_{Step} = \{\langle (q, q'), CFree \rangle\}$ is a single collision-free constraint. The transition relation does not exhibit any useful factoring. The initial clause is $\mathcal{C}_0 = \{\bar{x} = q^0\}$ and the goal clause is $\mathcal{C}_* = \{\bar{x} = q^*\}$. Figure 3 displays the constraint network for a plan skeleton of length k. Because the transition relation has a single clause, all solutions have this form. Dark gray parameters, such as the initial and final configurations, are constrained by constant equality. Free parameters are yellow.

Fig. 3. Motion planning plan skeleton of length k.

B. Pick-and-Place Planning

A pick-and-place domain is defined by a single robot with configuration space $\mathcal{Q} \subset \mathbb{R}^d$, a finite set of moveable objects $\mathcal{O}$, a set of stable placement poses $Stable_o \subset \text{SE}(3)$ for each object $o \in \mathcal{O}$, and a set of $Grasp_o \subset \text{SE}(3)$ relative grasp poses for each object $o \in \mathcal{O}$. The robot has a single manipulator that is able to rigidly attach itself to a single object at a time. The robot can execute trajectories τ specified by a sequence of configurations that respect joint limits and avoid fixed obstacles. We will assume that each trajectory also encodes the grasp of the object that the robot may be holding.

This domain can be modeled as a transition system with state-space $\bar{\mathcal{X}} = \mathcal{Q} \times \text{SE}(3)^{|\mathcal{O}|} \times (\{\textbf{None}\} \cup \mathcal{O})$. States are $\bar{x} = (q, p_1, ..., p_{|\mathcal{O}|}, h)$. Let $h \in \mathcal{O}$ indicate that the robot is holding object h and $h =$ **None** indicate that the robot's gripper is empty. When $h = o$, the pose p_o of object o is relative to the end-effector. Otherwise, p_o is relative to the environment. Controls are trajectories τ. The transition relation has $1 + 3|\mathcal{O}|$ clauses $\{\mathcal{C}_{Move}\} \cup \{\mathcal{C}^o_{MoveH}, \mathcal{C}^o_{Pick}, \mathcal{C}^o_{Place} \mid o \in \mathcal{O}\}$ because *pick*, *move-while-holding*, and *place* depend on o.

- $\mathcal{C}_{Move} = \{\langle (q, \tau, q'), Motion \rangle, h = \textbf{None}, h' = \textbf{None}\} \cup \{p_{o'} = p'_{o'}, \langle (\tau, p_{o'}), CFree_{o'} \rangle \mid o' \in \mathcal{O}\}$
- $\mathcal{C}^o_{MoveH} = \{h = o, h' = o, \langle (q, \tau, q', p_o), MotionH_i \rangle\} \cup \{p_{o'} = p'_{o'}, \langle (\tau, p_{o'}), CFree_{o'} \rangle \mid o' \in \mathcal{O}, o \neq o'\}$
- $\mathcal{C}^o_{Pick} = \{\langle (p_o), Stable_o \rangle, \langle (p'_o), Grasp_o \rangle, q = q', h = \textbf{None}, h' = o, \langle (p'_o, p_o, q), Kin_o \rangle\} \cup \{p_{o'} = p'_{o'} \mid o' \in \mathcal{O}, o \neq o'\}$
- $\mathcal{C}^o_{Place} = \{\langle (p_o), Grasp_o \rangle, \langle (p'_o), Stable_o \rangle, q = q', h = o, h' = \textbf{None}, \langle (p_o, p'_o, q), Kin_o \rangle\} \cup \{p_{o'} = p'_{o'} \mid o' \in \mathcal{O}, o \neq o'\}$

$Motion$ is the set of legal start configurations, trajectories, and end configurations. $MotionH_o$ is the set of legal start configurations, trajectories, end configurations, and grasps when holding object o. $CFree_o$ is the set of collision-free poses and trajectories with respect to object o. Kin_o is the set of kinematic solutions involving object o for a grasp, pose, and configuration.

Consider a pick-and-place problem with two movable objects A, B: initial state $\bar{x}^0 = (q^0, p^0_A, p^0_B, \textbf{None})$ is fully specified using equality constraints and $\bar{X}^*$ is given as constraints $\{\langle (p_A) \in Region \rangle, q = q^*\}$ where $Region \subseteq Stable_A$ is a region of poses. Figure 4 displays the constraint network for a plan skeleton that manipulates A and moves the robot to q^*. Thick edges indicate pairwise equality constraints. Light gray parameters are transitively fixed by pairwise equality. Despite having 29 total parameters, only 7 are free parameters. This highlights the strong impact of equality constraints on the dimensionality of the plan parameter space.

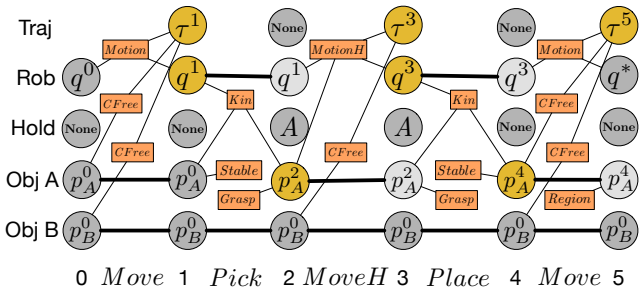

Fig. 4. Pick-and-place constraint network.

V. Sample-Based Planning

Constraints involving continuous variables are generally uncountably infinite sets, which are often difficult to characterize and reason with explicitly. Instead, each constraint can be described using a blackbox, implicit *test*. A test for constraint $C = \langle P, R \rangle$ is a boolean-valued function $f_C : \bar{\mathcal{Z}}_P \to \{0, 1\}$ where $f_C(\bar{z}_P) = [\bar{z}_P \in R]$. Implicit representations are used in sample-based motion planning, where they replace explicit representations of complicated robot and environment geometries with collision-checking procedures.

In order to use tests, we need to produce potentially satisfying values for $\bar{z}_P = (z_{p_1}, ..., z_{p_k})$ by sampling $\mathcal{Z}_{p_1}, ..., \mathcal{Z}_{p_k}$. Thus we still require an explicit representation for $\mathcal{X}_1, ..., \mathcal{X}_m$ and $\mathcal{U}_1, ..., \mathcal{U}_n$; however, these are typically less difficult to characterize. We will assume $\mathcal{X}_1, ..., \mathcal{X}_m$, $\mathcal{U}_1, ..., \mathcal{U}_n$ are each subsets of a bounded manifold. This strategy of sampling variable domains and testing constraints is the basis of *sample-based planning* [22]. These methods draw values from $\mathcal{X}_1, ..., \mathcal{X}_m$ and $\mathcal{U}_1, ..., \mathcal{U}_n$ using deterministic or random *samplers* for each space and test which combinations of sampled values satisfy required constraints.

Sample-based techniques are usually not complete over all problem instances. First, they cannot generally identify and terminate on infeasible instances. Second, they are often unable to find solutions to instances that require identifying values from a set that has very small or even zero measure in the space from which samples are being drawn (this is referred to as the "narrow passage" problem in motion planning). Thus, sample-based algorithms are typically only complete over *robustly feasibly* problems. A problem is robustly feasible if there exists a plan skeleton $\vec{a}$ such that $\mu(\bigcap_{C \in \mathcal{C}_{\vec{a}}} \widehat{C}) > 0$ where μ is a product measure on the plan-parameter space $\bar{\mathcal{Z}}$.

A. Dimensionality-reducing constraints

Some domains involve constraints that only admit a set of values on a lower dimensional subset of its parameter space. A *dimensionality-reducing constraint* C is one in which $\mu(\widehat{C}) = 0$ for all problems in the domain. Consider the *Stable* constraint. The set of satisfying values lies on a 3-dimensional manifold. By our current definition, all plans involving this constraint are not robustly feasible. When a problem involves dimensionality-reducing constraints, we have no choice but to sample at their intersection. This, in general, requires an explicit characterization of their intersection, which we may not have. Moreover, the number of dimensionality-reducing constraint combinations can be unbounded. However, for some domains, we can produce this intersection automatically using explicit characterizations for only a few spaces.

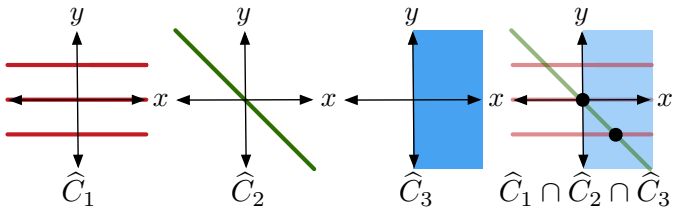

Fig. 5. Intersection of two dimensionality-reducing constraints.

We motivate these ideas with an example. Consider a plan skeleton $\vec{a}$ with parameters $(x, y) \in \bar{\mathcal{Z}} = [-2, +2]^2$ and constraints $\mathcal{C}_{\vec{a}} = \{C_1, C_2, C_3\}$ where $C_1 = \langle (y), \{-1, 0, 1\} \rangle$, $C_2 = \langle (x, y), \{(x, y) \mid x + y = 0\} \rangle$, and $C_3 = \langle (x), \{x \mid x \geq 0\} \rangle$. The set of solutions $\widehat{C}_1 \cap \widehat{C}_2 \cap \widehat{C}_3 = \{(1, -1), (0, 0)\}$ is 0-dimensional while the parameter space is 2-dimensional. This is because C_1 and C_2 are both dimensionality-reducing constraints. A uniform sampling strategy where $X, Y \sim \text{Uniform}(-2, +2)$ has zero probability of producing a solution. To solve this problem using a sampling-based approach, we must sample from $\widehat{C}_1 \cap \widehat{C}_2$. Suppose we are unable to analytically compute $\widehat{C}_1 \cap \widehat{C}_2$, but we do have explicit representations of C_1 and C_2 independently. In particular, suppose we know C_2 conditioned on values of y, $C_2(y) = \langle (x), \{-y\} \rangle$. Now, we can characterize $\widehat{C}_1 \cap \widehat{C}_2 = \{(x, y) \mid y \in R_1, x \in R_2(y)\} = \{(1, -1), (0, 0), (-1, 1)\}$. With respect to a counting measure on this discrete space, $\widehat{C}_1 \cap \widehat{C}_2 \cap \widehat{C}_3$ has positive measure. This not only gives a representation for the intersection but also suggests the following way to sample the intersection: $Y \sim \text{Uniform}(\{-1, 0, +1\})$, $X = -Y$, and reject (X, Y) that does not satisfy C_3. This strategy is not effective for all combinations of dimensionality-reducing constraints. Suppose that instead $C_1 = \langle (x, y), \{(x, y) \mid x - y = 0\} \rangle$. Because both constraints involve x and y, we are unable to condition on the value of one parameter to sample the other.

B. Intersection of Manifolds

In this section, we develop the topological tools to generalize the previous example. We start by defining conditional constraints, a binary partition of P for a constraint $\langle P, R \rangle$.

Definition 1. A *conditional constraint* $\langle I, O, R \rangle$ is given by a set of *input parameters* I, a set of *output parameters* O, and a relation R defined on $I \cup O$ where $I \cap O = \emptyset$.

In our analysis, we will assume all constraints are *constraint manifolds*. A constraint manifold $C = \langle P, M \rangle$ is a constraint where the relation is a manifold M defined by a finite set of charts. We will relate constraint manifolds back to constraints involving arbitrary relations in the subsequent section. The following lemma indicates that all conditional constraint manifolds are also manifolds when parameterized with values

for their input parameters. We give all proofs in supplementary material available here: http://web.mit.edu/caelan/www/publications/rss2017.pdf. Let $\text{proj}_P(\bar{Z}) = \{\bar{Z}_P \mid \bar{z} \in \bar{Z}\}$ be a set-theoretic projection of $\bar{Z}$ onto parameters P.

Lemma 1. *For any conditional constraint $\langle I, O, M\rangle$ where M is a d-dimensional manifold and for all $x \in \text{proj}_I(M)$, the set $\text{proj}_I^{-1}(x) = \{\bar{z} \in M \mid \bar{z}_I = x\}$ is a $(d - \dim \text{proj}_I(M))$-dimensional manifold.*

Let $S = \bigcap_{i=1}^{n} \widehat{M}_i$ be the intersection of constraint manifolds $\mathcal{M} = \{\langle P_1, M_1, \rangle, ..., \langle P_n, M_n\rangle\}$. $S \subseteq \bar{\mathcal{Z}}$ is defined on parameters $\Theta = \bigcup_{i=1}^{n} P_i$. We now present the main theorem which gives a sufficient condition for when S is a manifold. This theorem is useful because it identifies when the intersection of several possibly dimensionality-reducing constraints is a space that we can easily characterize.

Theorem 1. *S is a manifold of dimension $\sum_{i=1}^{n} \left(\dim M_i - \dim \text{proj}_{I_i}(M_i)\right)$ if there exists an ordering and conditioning of $\mathcal{M}$ into constraint manifolds $(\langle I_1, O_1, M_1\rangle, ..., \langle I_n, O_n, M_n\rangle)$ such that $\bigcup_{i=1}^{n} O_i = \Theta$ and $\forall i \in \{1, ..., n\}$:*

1) $\forall j \in \{1, ..., n\} \setminus \{i\}.\ O_i \cap O_j = \emptyset$,
2) $I_i \subseteq \bigcup_{j=1}^{i-1} O_j$
3) $\dim \text{proj}_{I_i}\left(\bigcap_{j=1}^{i} M_j\right) = \dim \text{proj}_{I_i}\left(\bigcap_{j=1}^{i-1} M_j\right)$

We will call S a *sample-space* when theorem 1 holds. From condition 1, each parameter must be the output of exactly one conditional constraint manifold. From condition 2, each input parameter must be an output parameter for some conditional constraint manifold earlier in the sequence. And from condition 3, the input parameter space $\text{proj}_{I_i}(M_i)$ must not reduce the dimensionality of the space. A sufficient and more direct criterion for condition 3 is that the input parameter space has full dimensionality.

Lemma 2. *If $\dim \text{proj}_{I_i}(M_i) = \dim \bar{\mathcal{Z}}_{I_i}$, then $\dim \text{proj}_{I_i}\left(\bigcap_{j=1}^{i} M_j\right) = \dim \text{proj}_{I_i}\left(\bigcap_{j=1}^{i-1} M_j\right)$*

Theorem 1 can be understood graphically using *sampling networks*. A sampling network is an acyclic orientation of a constraint network defined on constraint manifolds $\mathcal{M}$ in which each parameter node has exactly one incoming edge. Directed edges go from input parameters to constraints or constraints to output parameters. Each parameter is the output of exactly one constraint. Additionally, the graph is acyclic.

C. Robustness

When S is a sample-space, we can define a measure μ_S on it. Let μ_S be the uniform measure on the Euclidean codomain of S. Now we can provide a more general definition of robust feasibility. We will define robustness properties with respect to a set of constraint manifolds $\mathcal{M}$. This will allow us to analyze the set of solutions in a lower dimensional space where it may have nonzero measure.

Definition 2. A set of constraints $\mathcal{C}$ is *robustly satisfiable* with respect to $\mathcal{M}$ if for some subset of $\{\langle P_1, M_1, \rangle, ..., \langle P_n, M_n\rangle\} \subseteq \mathcal{M}$, their intersection $S = \bigcap_{i=1}^{n} \widehat{M}_i$ is a sample-space and $\mu_S(S \cap \bigcap_{C \in \mathcal{C}} \widehat{C}) > 0$.

Definition 3. A factored transition problem $\mathcal{P}$ is *robustly feasible* with respect to $\mathcal{M}$ if there exists a plan skeleton $\vec{a}$ such that $\mathcal{C}_{\vec{a}}$ is robustly satisfiable with respect to $\mathcal{M}$.

We still need to identify an appropriate set of constraint manifolds $\mathcal{M}$ for a domain. These are spaces within a domain for which we have an explicit representation. In particular, useful constraint manifolds are those that not only contain the low-dimensional intersection of one or more constraints but also have equivalent dimensionality. Thus, constraint manifolds can be thought of as a known, true space that a constraint resides in. More formally stated, a constraint manifold $\langle P, M\rangle$ is useful for a set of constraints $\{C_1, ..., C_k\}$ when $\bigcap_{i=1}^{k} \widehat{C}_i \subseteq M$ and for all other manifolds M' such that $\bigcap_{i=1}^{k} \widehat{C}_i \subseteq M', \dim M \leq \dim M'$.

Motion planning does not involve any dimensionality-reducing constraints. Thus, the configuration space itself is the only appropriate constraint manifold $\mathcal{M} = \{\langle (q), \mathcal{Q}\rangle\}$. In pick-and-place problems, *Stable*, *Region*, *Grasp*, *Kin*, *Motion*, and *MotionH* are all individually dimensionality-reducing constraints. Fortunately, we generally understand explicit representations of these sets barring collisions with fixed obstacles. In the subsequent section, we will show that these constraint manifolds are of sufficient dimension for many problems to be robustly feasible.

D. Conditional Samplers

Now that we have identified spaces that arise from dimensionality-reducing constraints, we can design samplers to draw values from these spaces. Our treatment of samplers will mirror the treatment of conditional constraints.

Definition 4. A *conditional sampler* ψ is a function from a set of input values $\bar{z}_I$ for input parameters I to a sampler. The sampler generates a sequence of output values $\psi(\bar{z}_I)$ for output parameters O using SAMPLE($\psi(\bar{z}_I)$).

We frequently design conditional samplers to intentionally draw values from conditional constraints. A useful conditional sampler for a kinematic constraint $\langle (g, q, p), Kin\rangle$ has input parameters $I = (g, p)$ and output parameters $O = (q)$. Conditional samplers can directly sample conditional constraints by performing rejection sampling on the conditional constraint manifold. For *Kin*, SAMPLE performs inverse kinematics, producing configurations q that have end-effector transform $g^{-1}p$. For a 7 degree-of-freedom manipulator in SE(3), this would sample from a 1-dimensional manifold.

A conditional sampler must generally produce values covering the constraint to allow completeness across a domain. In motion planning, a traditional sampler is *dense* with respect a topological space Z if the topological closure of its output sequence is Z. We extend this idea to conditional samplers.

Definition 5. A conditional sampler ψ is *dense* with respect to a conditional constraint $\langle I, O, R\rangle$ if $\forall \bar{z}_I \in \text{proj}_I(R)$, $\psi(z_I)$ is dense in $\text{proj}_O(\text{proj}_I^{-1}(\bar{z}_I))$.

Like conditional constraints, conditional samplers can be composed in a *sample sequence* $\vec{\psi} = (\psi_1, ..., \psi_k)$ to produce a vector of values for several parameters jointly. A well-formed sample sequence for a set of parameters Θ satisfies $\Theta = \bigcup_{j=1}^{n} O_j$ as well as conditions 1 and 2 from theorem 1. We are interested in identifying combinations of conditional samplers that will provably produce a solution for robustly feasible problems.

Definition 6. A set of conditional samplers $\Psi = \{\psi_1, ..., \psi_s\}$ is *sufficient* for a robustly satisfiable plan skeleton $\vec{a}$ with respect to $\mathcal{M}$ if there exists a sample sequence $\vec{\psi} \subseteq \Psi$ that with probability one samples a parameter assignment satisfying $\mathcal{C}_{\vec{a}}$ within a finite number of calls to SAMPLE.

Definition 7. Ψ is *sufficient* for a domain $\mathcal{D}$ with respect to $\mathcal{M}$ if for all robustly feasible $\mathcal{P} \in \mathcal{D}$, there exists a robustly satisfiable plan skeleton $\vec{a}$ for which Ψ is sufficient.

The following lemma indicates that dense conditional samplers for the appropriate conditional constraints will result in a sufficient collection of samplers.

Lemma 3. *A set of conditional samplers Ψ is sufficient for a robustly satisfiable plan skeleton $\vec{a}$ if for conditional constraint manifolds $(\langle I_1, O_1, M_1\rangle, ..., \langle I_n, O_n, M_n\rangle)$ satisfying theorem 1 and containing conditional constraints $(\langle I_1, O_1, R_1\rangle, ..., \langle I_n, O_n, R_n\rangle)$, $\forall i \in \{1, ..., n\}$, $\exists \psi_i \in \Psi$ that is dense with respect to $\langle I_i, O_i, R_i\rangle$.*

VI. Example Domains Revisited

Figure 6 shows a sampling network for the motion planning constraint network in figure 3. The green constraints are implicit domain constraints. A conditional sampler that has no inputs and is dense on $\mathcal{Q}$ is sufficient for this domain.

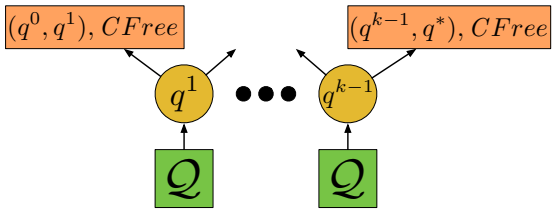

Fig. 6. Motion planning sampling network for constraint network in figure 3.

Figure 7 shows a sampling network for the pick-and-place constraint network in figure 4. The sampling network satisfies the graph theoretic conditions 1 and 2 in theorem 1. Additionally, each conditional constraint manifold (in red) has full dimensionality in its input parameter space. For Kin, a nonzero volume of poses and grasps admit an inverse kinematic solution. The same holds for $Motion$ and $MotionH$ but with respect to pairs of configurations and grasp transforms. Thus, the following set of conditional samplers is sufficient for this plan skeleton: $\langle (), (p), Region \cap Stable\rangle$, $\langle (), (p), Grasp\rangle$, $\langle (g, p), (q), Kin\rangle$, $\langle (q, q'), (\tau), Motion\rangle$, $\langle (q, q', g), (\tau), MotionH\rangle$.

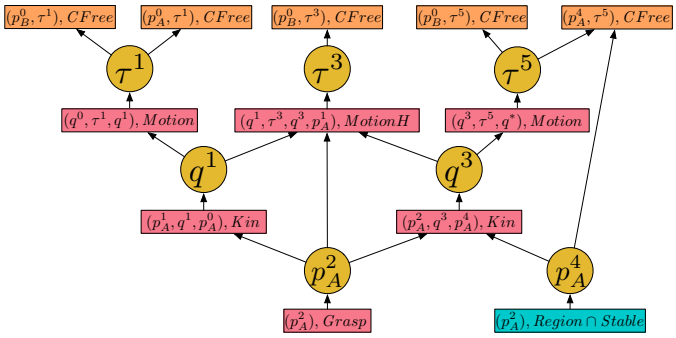

Fig. 7. Pick-and-place sampling network for constraint network in figure 4.

This sampling network structure generalizes to all pick-and-place problems with goal constraints on object poses and robot configurations. Each new cycle introduces a new grasp parameter, pose parameter, two configuration parameters, and two trajectory parameters. However, the only interaction with the next cycle is through the beginning and ending configurations which serve as the input parameters for the next move action. Thus, this small set of conditional samplers is enough to solve a large set of pick-and-place problems involving many objects.

VII. Algorithms

We have shown that, given a plan skeleton and sampling network, we can construct samplers that produce solutions. However, the input for a factored planning problem is just a transition system, initial set of states, and goal set of states. Thus, algorithms must search over plan skeletons and sampler sequences in order to produce solutions. Many skeletons will not admit solutions due to constraints absent from the sample-space, such as collision constraints, that are evaluated as tests. Like in sample-based motion planning, we are interested in identifying *probabilistically complete* algorithms.

Definition 8. An algorithm is *probabilistically complete* with respect to $\mathcal{D}$ and $\mathcal{M}$ if for all robustly feasible $\mathcal{P} \in \mathcal{D}$, it will return a plan in finite time with probability one.

We present algorithms that take, as a hyper-parameter input, a set of conditional samplers Ψ for the domain. The algorithms are therefore *domain-independent* because the problem-specific knowledge is restricted to the constraints and samplers. We will show that these algorithms are probabilistically complete, given a set of sufficient conditional samplers Ψ. Our Python implementation of the algorithms can be found here: https://github.com/caelan/stripstream.

We give two algorithms, INCREMENTAL and FOCUSED. Both algorithms share two common subroutines. First, SOLVE-DISCRETE constructs and solves a discretized transition systemffor problem $\mathcal{P} = \langle \mathcal{C}_0, \mathcal{C}_*, \{\mathcal{C}_1, ..., \mathcal{C}_\alpha\}\rangle$ given a set of samples from each $\mathcal{X}_i$ and $\mathcal{U}_i$. The procedure INSTANCES produces the set of states and transitions formed from samples that also satisfy their respective conjunctive constraint clauses $\mathcal{C}$. As a hyper-parameter, SOLVE-DISCRETE requires a blackbox search procedure SEARCH to find plans within

the discretized problem. This can be implemented using any sound and complete search algorithm such as breadth-first search (BFS). Artificial intelligence planning algorithms are much more efficient than classical graph search algorithms for high-dimensional, factored problems. Using generic heuristics, they can frequently avoid exploring most of the discrete state-space. Therefore, our implementation automatically compiles to Planning Domain Definition Language (PDDL) [30] in order to use the efficient FastDownward planning system [19].

```
SOLVE-DISCRETE(⟨C_0, C_*, {C_1, ..., C_α}⟩, samples | SEARCH):
1  initial = INSTANCES(C_0, samples); goal = INSTANCES(C_*, samples)
2  transitions = {INSTANCES(C, samples) for C in {C_1, ..., C_α}}
3  return SEARCH(initial, goal, transitions)

PROCESS(queue, samples, CALL, blocked, k):
1  processed = ∅
2  while len(queue) ≠ 0 and len(processed) < k :
3      ψ(inps) = POP(queue)
4      samples += CALL(ψ(inps)); processed += {ψ(inps)}
5      for ψ'(inps') in reduce(INSTANCES(ψ', samples) for ψ' in Ψ)
6          if ψ'(inps') not in (queue + processed + blocked):
7              PUSH(queue, ψ'(inps'))
8  return processed
```

Second, PROCESS iteratively calls *sampler instances*, conditional samplers Ψ conditioned on particular values. PROCESS's inputs are a $queue$ of samplers, a set of $samples$, and three additional parameters that are used differently by INCREMENTAL and FOCUSED. CALL is a procedure that takes as input a sampler instance and returns a set of values, $blocked$ is a set of samplers which are not to be processed, and k is the maximum number of iterations. On each iteration, PROCESS pops a sampler instance $\psi(inps)$ off of $queue$, adds the result of CALL to $samples$, and adds $\psi(inps)$ to $processed$, a set of processed samplers. New sampler instances $\psi'(inps')$ resulting from the produced values are added to $queue$. After k iterations or $queue$ is empty, PROCESS returns $processed$.

A. Incremental Algorithm

```
INCREMENTAL(P | Ψ, SEARCH):
1  samples = GET-SAMPLES(P)
2  queue = reduce(INSTANCES(ψ, samples) for ψ in Ψ)
3  while True:
4      processed = PROCESS(queue, samples, ∅ | SAMPLE, len(queue))
5      plan = SOLVE-DISCRETE(P, samples, SEARCH)
6      if plan ≠ None: return plan
7      PUSH(queue, processed)
```

The INCREMENTAL algorithm alternates between generating samples and checking whether the current set of samples admits a solution. It can be seen as a generalization of the probabilistic roadmap (PRM) [22] for motion planning and the FFRob algorithm for task and motion planning [16]. INCREMENTAL maintains a $queue$ of samplers. On each iteration, INCREMENTAL calls the PROCESS subroutine to sample at most **len**($queue$) samplers using the function SAMPLE. It calls SOLVE-DISCRETE to attempt to find a $plan$ using the current set of $samples$. If SOLVE-DISCRETE is successful, $plan$ is returned. Otherwise, INCREMENTAL adds the $processed$ sampler instances back to $queue$ to be used again on later iterations.

Theorem 2. INCREMENTAL *is probabilistically complete for a domain given a sufficient set of conditional samplers.*

Because INCREMENTAL creates sampler instances exhaustively, it will produce many unnecessary samples.

B. Focused Algorithm

The FOCUSED algorithm uses *lazy samples* as placeholders for actual concrete sample values. Lazy samples are similar in spirit to symbolic references [34]. The lazy samples are optimistically assumed to satisfy constraints with concrete samples and other lazy samples. This allows SOLVE-DISCRETE to reason about plan skeletons without some concrete parameters. After finding a plan, FOCUSED calls samplers that can produce values for the lazy samples used. This algorithm is related to a lazy PRM [4, 9], which defers collision checks until a path is found. However, FOCUSED defers generation of entire samples until an optimistic plan is found.

```
LAZY-CALL(ψ(inps)):
1  return [LAZYSAMPLE(ψ(inps), out) for out in OUTPUTS(ψ)]
```

On each iteration, the FOCUSED algorithm creates a new $queue$ and calls PROCESS to produce $mixed\_samples$. It passes the procedure LAZY-CALL rather than SAMPLE in to PROCESS. For each output out of ψ, LAZY-CALL creates a unique object for the combination of ψ, $inps$, and out. The inputs inp may be lazy samples themselves. In order to avoid producing an infinite number of lazy samples, out becomes shared across sampler inputs after a fixed depth.

```
RETRACE(sample):
1  if not IS-LAZY(sample): return ∅
2  ψ(inps) = GET-SAMPLER(sample)
3  return reduce(RETRACE(inp) for inp in inps) + {ψ(inps)}

FOCUSED(P | Ψ, SEARCH):
 1  samples = GET-SAMPLES(P); new_samples = ∅; called = ∅
 2  while True:
 3      mixed_samples = COPY(samples)
 4      queue = reduce(INSTANCES(ψ, mixed_samples) for ψ in Ψ)
 5      PROCESS(queue, mixed_samples, LAZY-CALL, called, ∞)
 6      plan = SOLVE-DISCRETE(P, mixed_samples | SEARCH)
 7      if plan = None:
 8          samples += new_samples; new_samples = ∅; called = ∅
 9          continue
10      if GET-SAMPLES(plan) ⊆ samples: return plan
11      for ψ(inps) in reduce(RETRACE(s) for s in GET-SAMPLES(plan)):
12          if inps ⊆ samples:
13              new_samples += SAMPLE(ψ, inps); called += {ψ(inps)}
```

SOLVE-DISCRETE performs its discrete search using $mixed\_samples$, a mixed set of samples and lazy samples. If SOLVE-DISCRETE returns a $plan$, FOCUSED first checks whether it does not use any lazy samples, in which case it returns $plan$. Otherwise, it calls RETRACE to extract the set of sampler instances used to produce the lazy samples. For each sampler instance ψ(*inps*) without lazy sample inputs, FOCUSED draws a sample and adds it to *new_samples*. To ensure all relevant samplers are called, each sampler instance is added to *called* after it is processed. This prevents these sampler instances from constructing lazy samples within PROCESS. Additionally, samples are added to $new\_samples$ before they are moved to $samples$ to limit the growth in sampler instances.

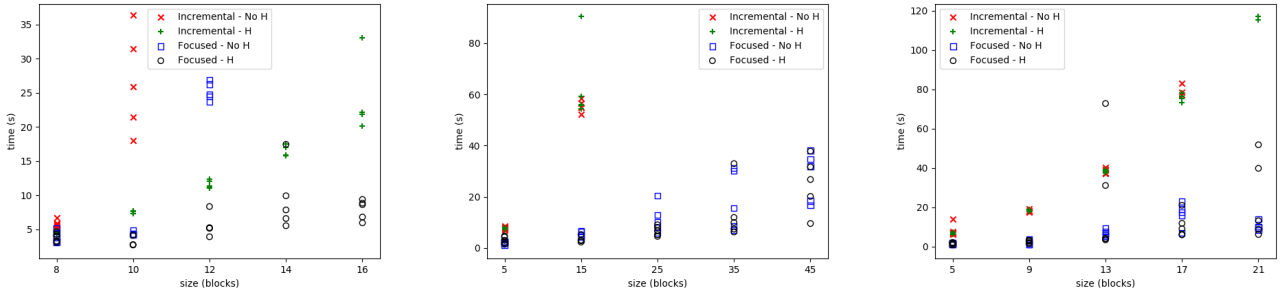

Fig. 8. Total runtime of the algorithms over 5 trials per problem size. Left: experiment 1. Center: experiment 2. Right: experiment 3

When SOLVE-DISCRETE fails to find a plan, *new_samples* are added to *samples*, *called* is reset, and the process repeats. In practice, to bias SEARCH to use few lazy samples, we add a non-negative cost to each transition instance corresponding to the number of lazy samples used.

Theorem 3. FOCUSED *is probabilistically complete for a domain given a sufficient set of conditional samplers.*

VIII. Experiments

We performed three scaling experiments on pick-and-place problems. All experiments used the same factored transition system and conditional samplers. The conditional samplers for poses, grasps, inverse kinematics, and motion plans were implemented using OpenRAVE [11]. All experiments considered five problem sizes, varying the number of objects. We considered two FastDownward configurations for the INCREMENTAL and FOCUSED algorithms: *H* uses the FastForward heuristic [20] in a greedy search and *No H* does not. Both configurations benefit from a compilation process that can quickly detect some infeasible problems using admissible heuristics. We performed five trials using randomly (with the exception of experiment 2) generated problem instances for each problem size. All trials were run on 2.8 GHz Intel Core i7 processor with a 120 second time limit. The scatter plots in figure 9 display the total runtime of each configuration per trial. Timeouts are indicated by the omission of a trial.

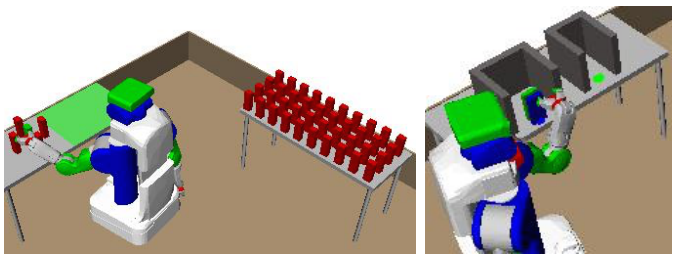

Fig. 9. Left: experiment 2. Right: regrasp problem.

Experiment 1 in figure 9 is the "grid@tabletop" benchmark [24] where each object has a specified goal pose. The initial placements are randomly generated. The table size scales with the number of objects. Each object has a single top-grasp. *Focused - H* solved all problem instances and *Incremental - H* solved all but one (size=14) indicating that use of a heuristic is necessary for problems with long-horizons.

Experiment 2 in figure 9 has the goal that a single green object be placed in the green region. The green object is obstructed by four red objects. The number of distracting red objects on the right table is varied between 0 and 40. Each object has four side-grasps. This experiment reflects many real-world environments where the state-space is enormous but many objects do not substantially affect a task. Both *Focused - No H* and *Focused - H* solved all problem instances showing that the FOCUSED algorithm is able to avoid producing samples for objects until they are relevant to the task.

Experiment 3 in figure 1 has the goal that a single blue object be moved to a different table. The blue object starts at the center of the visible table, and the red objects are randomly placed on the table. The table size scales with the number of objects. Each object has four side-grasps. *Focused - H* solved all instances and *Focused - No H* solved all but one (size=21).

We also performed five trials on the non-monotonic, regrasp problem in figure 9. The goal constraints are that the green object be at the green pose and the blue object remain at its current pose. The robot must place the green object at an intermediate pose to obtain a new grasp in order to insert it in the thin, right cupboard. All configurations solved all trials in less than 5 seconds. This indicates that the algorithms can solve pick-and-place problems where even non-collision constraints affect the plan skeleton of solutions.

IX. Conclusion

We introduced the idea of conditional constraints and samplers. This allowed us to give a general definition of robust feasibility for factored transition systems. We gave two general-purpose algorithms that are probabilistically complete given sufficient samplers. We demonstrated that these algorithms are effective at solving challenging pick-and-place problems.

X. Acknowledgements

We acknowledge support from NSF grants 1122374, 1420927, and 1523767, ONR grant N00014-14-1-0486, and ARO grant W911NF1410433. Any opinions, findings, and conclusions or recommendations expressed are our own and do not necessarily reflect the views of our sponsors.

References

[1] R. Alami, J.-P. Laumond, and T. Siméon. Two manipulation planning algorithms. In *Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1994. URL http://dl.acm.org/citation.cfm?id=215085.

[2] Rachid Alami, Thierry Siméon, and Jean-Paul Laumond. A geometrical approach to planning manipulation tasks. the case of discrete placements and grasps. In *International Symposium of Robotic Research (ISRR)*, 1990. URL http://dl.acm.org/citation.cfm?id=112736.

[3] Jennifer Barry, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. A hierarchical approach to manipulation with diverse actions. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1799–1806. IEEE, 2013. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.365.1060.

[4] Robert Bohlin and Lydia E Kavraki. Path planning using lazy PRM. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 521–528. IEEE, 2000. URL http://ieeexplore.ieee.org/document/844107/.

[5] Stephane Cambon, Rachid Alami, and Fabien Gravot. A hybrid approach to intricate motion, manipulation and task planning. *International Journal of Robotics Research (IJRR)*, 28, 2009. URL http://journals.sagepub.com/doi/abs/10.1177/0278364908097884.

[6] Neil T. Dantam, Z. Kingston, Swarat Chaudhuri, and Lydia E. Kavraki. Incremental task and motion planning: A constraint-based approach. In *Robotics: Science and Systems (RSS)*, 2016. URL http://www.roboticsproceedings.org/rss12/p02.pdf.

[7] Lavidra de Silva, Amit Kumar Pandey, Mamoun Gharbi, and Rachd Alami. Towards combining HTN planning and geometric task planning. In *RSS Workshop on Combined Robot Motion Planning and AI Planning for Practical Applications*, 2013. URL https://arxiv.org/abs/1307.1482.

[8] Rina Dechter. Constraint networks. Technical report, Information and Computer Science, University of California, Irvine, 1992. URL http://www.ics.uci.edu/~csp/r17-survey.pdf.

[9] Christopher M Dellin and Siddhartha S Srinivasa. A unifying formalism for shortest path problems with expensive edge evaluations via lazy best-first search over paths with edge selectors. *International Conference on Automated Planning and Scheduling (ICAPS)*, 2016. URL https://arxiv.org/abs/1603.03490.

[10] Ashwin Deshpande, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Decidability of semi-holonomic prehensile task and motion planning. *Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2016. URL http://lis.csail.mit.edu/pubs/deshpande-WAFR16.pdf.

[11] Rosen Diankov and James Kuffner. Openrave: A planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute, Carnegie Mellon University, 2008. URL https://pdfs.semanticscholar.org/c28d/3dc33b629916a306cc58cbff05dcd632d42d.pdf.

[12] Christian Dornhege, Patrick Eyerich, Thomas Keller, Sebastian Trüg, Michael Brenner, and Bernhard Nebel. Semantic attachments for domain-independent planning systems. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 114–121. AAAI Press, 2009. URL https://www.aaai.org/ocs/index.php/ICAPS/ICAPS09/paper/viewPaper/754.

[13] Christian Dornhege, Andreas Hertle, and Bernhard Nebel. Lazy evaluation and subsumption caching for search-based integrated task and motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Workshop on AI-based robotics*, 2013. URL https://robohow.eu/_media/workshops/ai-based-robotics-iros-2013/paper08-final.pdf.

[14] 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 (WAFR)*, 2014. URL https://link.springer.com/chapter/10.1007%2F978-3-319-16595-0_11.

[15] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Backward-forward search for manipulation planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015. URL http://lis.csail.mit.edu/pubs/garrett-iros15.pdf.

[16] Caelan Reed Garrett, Tomas Lozano-Perez, and Leslie Pack Kaelbling. FFRob: Leveraging symbolic planning for efficient task and motion planning. *arXiv preprint arXiv:1608.01335*, 2016. URL https://arxiv.org/abs/1608.01335.

[17] K. Hauser and J.C. Latombe. Integrating task and prm motion planning: Dealing with many infeasible motion planning queries. In *International Conference on Automated Planning and Scheduling (ICAPS) Workshop on Bridging the Gap between Task and Motion Planning*, 2009.

[18] Kris Hauser and Victor Ng-Thow-Hing. Randomized multi-modal motion planning for a humanoid robot manipulation task. *International Journal of Robotics Research (IJRR)*, 30(6):676–698, 2011. URL http://journals.sagepub.com/doi/abs/10.1177/0278364910386985.

[19] Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research (JAIR)*, 26:191–246, 2006. URL http://www.jair.org/papers/paper1705.html.

[20] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal Artificial Intelligence Research (JAIR)*, 14:253–302, 2001. URL http://dl.acm.org/citation.cfm?id=1622404.

[21] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical planning in the now. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

URL http://ieeexplore.ieee.org/document/5980391/.

[22] 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*, 12(4):566–580, 1996. URL http://ieeexplore.ieee.org/document/508439/.

[23] A. Krontiris and K. E. Bekris. Dealing with difficult instances of object rearrangement. In *Robotics: Science and Systems (RSS)*, Rome, Italy, 07/2015 2015. URL http://www.cs.rutgers.edu/~kb572/pubs/Krontiris_Bekris_rearrangement_RSS2015.pdf.

[24] A. Krontiris and K. E. Bekris. Efficiently solving general rearrangement tasks: A fast extension primitive for an incremental sampling-based planner. In *International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, 05/2016 2016. URL http://www.cs.rutgers.edu/~kb572/pubs/fast_object_rearrangement.pdf.

[25] Fabien Lagriffoul, Dimitar Dimitrov, Alessandro Saffiotti, and Lars Karlsson. Constraint propagation on interval bounds for dealing with geometric backtracking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012. URL http://ieeexplore.ieee.org/document/6385972/.

[26] Fabien Lagriffoul, Dimitar Dimitrov, Julien Bidot, Alessandro Saffiotti, and Lars Karlsson. Efficiently combining task and motion planning using geometric constraints. *International Journal of Robotics Research (IJRR)*, page 0278364914545811, 2014. URL http://journals.sagepub.com/doi/abs/10.1177/0278364914545811?journalCode=ijra.

[27] T. Lozano-Pérez, J. L. Jones, E. Mazer, P. A. O'Donnell, W. E. L. Grimson, P. Tournassoud, and A. Lanusse. Handey: A robot system that recognizes, plans, and manipulates. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1987. URL http://ieeexplore.ieee.org/document/1087847/.

[28] Tomás Lozano-Pérez. Automatic planning of manipulator transfer movements. *IEEE Transactions on Systems, Man, and Cybernetics*, 11:681–698, 1981. URL http://ieeexplore.ieee.org/document/4308589/.

[29] Tomás Lozano-Pérez and Leslie Pack Kaelbling. A constraint-based method for solving sequential manipulation planning problems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3684–3691. IEEE, 2014. URL http://lis.csail.mit.edu/pubs/tlpk-iros14.pdf.

[30] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. Pddl: The planning domain definition language. Technical report, Yale Center for Computational Vision and Control, 1998. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.9941.

[31] Amit Kumar Pandey, Jean-Philippe Saut, Daniel Sidobre, and Rachid Alami. Towards planning human-robot interactive manipulation tasks: Task dependent and human oriented autonomous selection of grasp and placement. In *RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*, 2012. URL http://ieeexplore.ieee.org/abstract/document/6290776/.

[32] Erion Plaku and Gregory Hager. Sampling-based motion planning with symbolic, geometric, and differential constraints. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010. URL http://ieeexplore.ieee.org/document/5509563/.

[33] Thierry Siméon, Jean-Paul Laumond, Juan Cortés, and Anis Sahbani. Manipulation planning with probabilistic roadmaps. *International Journal of Robotics Research (IJRR)*, 23(7-8):729–746, 2004. URL http://journals.sagepub.com/doi/abs/10.1177/0278364904045471.

[34] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2014. URL http://ieeexplore.ieee.org/document/6906922/.

[35] Mike Stilman and James J. Kuffner. Planning among movable obstacles with artificial constraints. In *Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2006.

[36] Mike Stilman, Jan-Ulrich Schamburek, James J. Kuffner, and Tamim Asfour. Manipulation planning among movable obstacles. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2007. URL http://ieeexplore.ieee.org/document/4209604/.

[37] Marc Toussaint. Logic-geometric programming: an optimization-based approach to combined task and motion planning. In *AAAI Conference on Artificial Intelligence*, pages 1930–1936. AAAI Press, 2015. URL https://www.ijcai.org/Proceedings/15/Papers/274.pdf.

[38] Jur Van Den Berg, Mike Stilman, James Kuffner, Ming Lin, and Dinesh Manocha. Path planning among movable obstacles: a probabilistically complete approach. In *Algorithmic Foundation of Robotics VIII*, pages 599–614. Springer, 2009. URL https://link.springer.com/chapter/10.1007%2F978-3-642-00312-7_37.

[39] William Vega-Brown and Nicholas Roy. Asymptotically optimal planning under piecewise-analytic constraints. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016. URL http://www.wafr.org/papers/WAFR_2016_paper_11.pdf.

[40] Gordon T. Wilfong. Motion planning in the presence of movable obstacles. In *Symposium on Computational Geometry*, pages 279–288, 1988. URL https://link.springer.com/article/10.1007/BF01530890.