Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning

Marc Toussaint
Machine Learning & Robotics Lab
University Stuttgart, Germany
marc.toussaint@ipvs.uni-stuttgart.de

Abstract

We consider problems of sequential robot manipulation (aka. combined task and motion planning) where the objective is primarily given in terms of a cost function over the final geometric state, rather than a symbolic goal description. In this case we should leverage optimization methods to inform search over potential action sequences. We propose to formulate the problem holistically as a 1st-order logic extension of a mathematical program: a non-linear constrained program over the full world trajectory where the symbolic state-action sequence defines the (in-)equality constraints. We tackle the challenge of solving such programs by proposing three levels of approximation: The coarsest level introduces the concept of the effective end state kinematics, parametrically describing all possible end state configurations conditional to a given symbolic action sequence. Optimization on this level is fast and can inform symbolic search. The other two levels optimize over interaction keyframes and eventually over the full world trajectory across interactions. We demonstrate the approach on a problem of maximizing the height of a physically stable construction from an assortment of boards, cylinders and blocks.

1 Introduction

Consider the following problem: "Given an assortment of boards and cubic and cylindrical objects, and a cup, maximize the height of the cup in the final configuration. You are allowed any sequence of interactions with the environment, but eventually the cup should be placed stably without having it in the hand in the final configuration." A solution to this task might be to build a high, stable construction from the given objects and place the cup on top. Or simply place it on a high cabinet, in case there is one around. But crucially, the objective is given only in terms of an evaluation function of the final configuration and potential control costs.

Solving such tasks requires an integrated approach to joint logic and geometric reasoning—which we think is a core challenge in the intersection of AI and robotics. There recently have been very impressive advances on combined task and motion planning (TAMP), which we review below. For instance, Srivastava et al. [2014] have proposed a general approach for interweaving existing symbolic planners (e.g., for tasks describable in PDDL) with existing geometric planners (e.g., RRTs) to check the geometric feasibility of the actions proposed by preliminary symbolic plans.

The specific methods we propose in this paper are (yet!) in many respects less efficient than existing TAMP approaches; in particular we cannot scale to similar depths (action sequence horizons) of symbolic planning. However, it would be hard to make existing TAMP methods fit to solve the tasks mentioned above. First, we aim for planners that can deal with arbitrary objective functions $\psi(x(T))$ on the final geometric configuration $x(T)$ and overall control costs. In special cases it might be possible for the human expert to introduce symbols to reflect the objective; but we want to avoid this. Most existing TAMP approaches, however, require a well-defined task planning problem including a symbolic goal description. Further, the above problems are inherently optimization problems, not constraint satisfaction problems.

In this paper we therefore propose an approach that roots in a mathematical programming formulation. We think of sequential manipulation as a special kind of logic-geometric program (LGP). Let us first sketch what we mean by an LGP in general: Given are a logic $\mathcal{L}$, a knowledge base $K \in \mathcal{L}$, and an objective function $f(x)$ over a continuous object $x \in \mathcal{X}$. Assume that any $\alpha \in \mathcal{L}$ defines constraint functions $g(x|\alpha) : \mathcal{X} \to \mathbb{R}^{d(\alpha)}$ and $h(x|\alpha) : \mathcal{X} \to \mathbb{R}^{e(\alpha)}$, where the number of constraints (dimensionalities $d(\alpha)$ and $e(\alpha)$ of g and h, respectively) also depend on α. Then a logic-geometric program[1] is

$$\min_{x,\alpha} f(x,\alpha) \quad \text{s.t.} \quad \alpha \models K,\ g(x,\alpha) \le 0,\ h(x,\alpha) = 0 \quad (1)$$

In the case of manipulation, x will be the full path of all objects and the robot across object interactions. The role of the symbol α is to define geometric and differential constraints on this path, e.g., constraints implied by the symbolic decision to make an object movable (grasp it). In this way our approach utilizes the symbolic level to structure the optimization over paths of all involved objects—in terms of the constraints that symbolic decisions imply within the mathematical program.

[1] We could actually drop calling it "geometric", but want to express our hope that it is particularly useful for geometric reasoning.

Thereby, in contrast to previous TAMP approaches to robot manipulation our formulation provides a notion of global optimality over the full (logic and geometric) sequential manipulation.

Clearly, computing a globally optimal solution to the program will be infeasible. The most interesting research questions concern how to tackle it in a feasible way. In this paper we propose three levels on which geometric reasoning (that is, optimization over geometric configurations and paths) may inform symbolic search towards a joint optimum. All three levels raise novel interesting challenges for motion (or configuration) optimizers. The highest level, which plays the crucial role of the heuristic for informed search, is perhaps most interesting and will optimize over the end configuration *conditional to a symbolic action sequence*. For this, we have to analyze the space of possible object configurations that can be reached with a given symbolic action sequence (an action skeleton in the sense of [Lozano-Pérez and Kaelbling, 2014]), leaving free the geometric parameters of all actions. We call this the *effective end space*. Optimizing over this space is a form of geometric reasoning which can account for any geometric features that define $\psi(x(T))$, including for instance the stability of a construction. Roughly, it "moves" the objects in the final construction into place to maximize (e.g.) stability—and thereby informs also the early actions in the action sequence on how to place objects to account for the temporally delayed effect of stability of the final construction. We think that this kind of "backward regression" of geometric information is non-existent in the planners of previous formulations. Besides the novel formulation of manipulation planning as an LGP, we think of the concept of the effective end space and its optimization as a search heuristic as the core contributions of this paper. The other two levels of exploiting geometric optimization concern (approximate) path optimization. This implies the challenge of motion optimization across kinematic switches of the world configuration (across action boundaries) to allow for the optimization over the full manipulation sequence. We detail our contributions on this later.

In the following section we discuss related work. Section 3 introduces to the problem formalization. Section 4 discusses our proposed solver, and section 5 experiments on building a stable construction that is as high as possible.

2 Related Work

2.1 Combined Task and Motion Planning

We first focus on TAMP approaches to robot manipulation. The approach of Lozano-Pérez and Kaelbling [2014] is strongly related to ours, but eventually aims to reduce the problem to a CSP instead of a continuous mathematical programm. They introduce the notion of an action *skeleton*, which is a sequence of symbolic actions leaving their specific geometric parameters unspecified. As in our approach, the decisions on the geometric parameterization of actions are *deferred* to a stage where the skeleton has been fixed. However, they argue that "The constraints from a plan skeleton are significantly nonlinear and much too complex to solve exactly analytically in continuous form". While we agree that an *analytic* solution is out of reach, it is exactly the aim of this paper to formulate the problem as a continuous mathematical program that can be solved (locally) optimally using constrained optimization methods. Lozano-Pérez and Kaelbling [2014] rely on discretizing the geometric parameters of a skeleton.

The approach [Lozano-Pérez and Kaelbling, 2014] is related to [Lagriffoul *et al.*, 2012; 2014], who also employ CSP methods (constraint propagation) to resolve geometric constraints. Siméon et al. [2004] describe complex, multi-interaction planning of the manipulation of a single object, but does not bridge to relational/logic representations of environments with many objects.

Garrett et al. [2014] addresses the problem in a way that uses symbols (`CanGrasp` and `Reachable`) to represent geometric preconditions for actions also on the symbolic level. These predicates depend on the actual geometric configuration; their approach again samples configurations to discretize such continuous variables. Similar to this, Srivastava et al. [2014] devise a symbolic description that includes predicates to abstract geometric feasibility conditions and represent action operator preconditions on the symbolic level. For a given task plan, the predicates are evaluated on demand, as well as the `Obstructs` predicate added depending on which objects make a path finder fail. Similar kind of backtracking depending on geometrical reasoning is also the core idea in [Pandey *et al.*, 2012; de Silva *et al.*, 2013; Alili *et al.*, 2010]. Other non-optimization based planners are [Sucan and Kavraki, 2011], representing motion options in a task motion multigraph, and [Plaku and Hager, 2010], sampling-based motion and symbolic action planning under differential constraints.

In summary, these TAMP approaches to robot manipulation assume a symbolic goal description in their demonstrations. None explicitly aims to optimize a final configuration that is evaluated only by an objective function.

From a higher level perspective, previous TAMP approaches to robot manipulation typically introduce predicates to represent geometric conditions or features (including the existence of feasible paths). These are necessary as preconditions of action operators. The evaluation of these predicates (on demand) then calls geometric motion planning methods like subroutines. The general idea to introduce the "right" symbolic abstractions of geometry such that one can then reason *only* on the symbolic level (modulo calling subroutines to evaluate the symbols) is intriguing, as symbolic reasoning seems so much more efficient than geometric. But this implies the fundamental and long-standing question of what are the right symbolic abstractions. Instead of trying to represent geometric constraints (or other geometric aspects) on the symbolic level, the logic-geometric programming approach aims to tackle optimization problems directly on the geometric level, where the role of logic is to control the constraints in the mathematical program. Expressed differently: *instead of trying to pull geometry into logic representations, we try to pull logic into mathematical programming*.

Outside the domain of robot manipulation, the Kingming planner of Li and Williams [2008] reduces a mission planning problem for an autonomous unverwater vehicle to a mixed logic nonlinear program (MLNLP). This approach in-

deed proceeds ours in using logic in a mathematical programming formulation of the mission problem. However, the approach is not applicable to object manipulation and movable objects. Further Ivankovic et al. [2014] formulate, e.g., power supply restoration problems in a way where symolic variables imply constraints on continuous optimization problems, as is the case in ours.

2.2 Trajectory Optimization Across Manipulation

[Mordatch *et al.*, 2012] presented an impressive method for optimizing full manipulation trajectories, based on a contact-invariant optimization approach that originated in locomotion research. The resulting trajectories are very smooth (though contact constraints of grasps seem not accurately fulfilled). The approach however relies on rather precise descriptions of the desired movement of objects; task planning, or task-motion planning that optimizes over features of the final configuration seem out of scope for this method. Equally impressive, but not aiming at sequential manipulation planning, are recent methods on trajectory optimization through contacts [Posa *et al.*, 2014]. This paper also extensively discusses related literature from the area of locomotion, which is an area that early on had to deal with trajectory optimization through kinematic switches of the configuration.

Finally, let us mention qualitative state plans [Hofmann and Williams, 2006] as an approach that also aims to reduce sequential task planning and control to mathematical programming. However, to our knowledge these have not yet been applied to object manipulation tasks.

3 Problem Formulation

We first introduce the specific problem formulation addressed in this paper and below discuss in what sense this is a special case of what we consider a logic-geometric program. Let $x: [0,T] \to \mathcal{X}$ be a path in the configuration space $\mathcal{X}$ of the whole environment, including the robot and all object configurations. We can explicate $\mathcal{X} = \mathbb{R}^n \times SE(3)^m$ for an n-dimensional robot interacting with m objects—but we will never need to parameterize this full space explicitly. We assume the initial configuration $x(0)$ given. Complementary to the path, the problem formulation is in terms of a first-order logic $\mathcal{L}$, an initial state $s_0 \in \mathcal{L}$, and a sequence of T action operators that transition the logic state as $s_k = \mathit{succ}(a_k, s_{k\text{-}1})$. Further, let $t_k \in [0,T], k = 1,..,K$ be K points in time. We consider the program

$$\min_{x, a_{1:K}, s_{1:K}, t_{1:K}} \int_0^T c(x(t), \dot{x}(t), \ddot{x}(t))\, dt + \psi(x(T)) \quad (2)$$

$$\text{s.t.} \quad \forall_{k=1:K}\ s_k = \mathit{succ}(a_k, s_{k\text{-}1}) \quad (3)$$

$$s_K \models \mathfrak{g} \quad (4)$$

$$\forall_{k=1}^K\ h_{switch}(x(t_k) \,|\, a_k, s_{k\text{-}1}) = 0$$
$$\forall_{k=1}^K\ g_{switch}(x(t_k) \,|\, a_k, s_{k\text{-}1}) \le 0 \quad (5)$$

$$\forall_{t\in[0,T]}\ h_{path}(x(t), \dot{x}(t) \,|\, s_{k(t)}) = 0$$
$$\forall_{t\in[0,T]}\ g_{path}(x(t), \dot{x}(t) \,|\, s_{k(t)}) \le 0\,. \quad (6)$$

Eq. (2) is a typical optimal control objective on a path, with running costs c and an evaluation ψ of the final configuration. However, this optimization is w.r.t. a series of equality and inequality constraints. It is the role of the logic to define these constraints on the path. The constraints (5) concern the consistency of the switch configuration $x(t_k)$ with the action operator a_k. The constraints (6) imply (geometric & differential) constraints on the path $x(t)$ depending on the current symbolic state $s_{k(t)}$, where $k(t) = \max\{k : t_k \le t\}$ (such that $t \in [t_{k(t)}, t_{k(t)+1}]$) and we assume that time points t_k increase with the step count k.

Recall that, in the general context of classical mechanics, the term *kinematics* refers to the description of the possible motions of a system. In that sense, equation (6) says that s_k defines the path kinematics in the interval $[t_k, t_{k+1}]$. Let us give an example: consider an n-dimensional robot[2] in an environment with m objects. At any point in time, the world configuration kinematics will only span an n-dimensional space. However, which degrees of the world configuration are actually actuated crucially depends on what the robot is interacting with, whether it has an object `inhand` ($\rightsquigarrow$ pose equality constraint), `pushes` a drawer with a finger tip ($\rightsquigarrow$ ball-joint equality constraint and velocity inequality constraint), etc. Our approach of problem formulation fundamentally implies that symbols need to represent exactly these categorial aspects of the world configuration kinematics in equation (6): *State symbols define kinematics*.

If state symbols fundamentally define the system kinematics, then *action operators correspond to kinematic switches*. Examples are the establishment of a grasp, or a push contact, or the releasing of an object. This clarifies why we call $x(t_k)$ a switch configuration. The constraints (5) need to ensure geometric consistency with the assumption that $x(t_k)$ describes a grasp pose, push contact, or object release.

The fact that the world configuration kinematics are always only n-dimensional also allows us to represent the path $x(t)$ only as an n-dimensional path—but together with a description of the configuration kinematics, including the (fixed) relative transformations between objects that are implicit in the kinematic switches. The following sections will explain this in more detail.

Finally, equations (3) requires that a_k are action operators that generate a sequence of states $s_k \in \mathcal{L}$ and, optionally, (4) requires that the final state entails a symbolic goal. Such given symbolic goals are the standard in previous approaches to integrated task and motion planning. We will not consider such a symbolic goal and instead focus on how to plan when "goal information" is implicit only in the evaluation of the final world configuration.

4 Solver: Boiling Down the Problem on Three Levels

Our general approach is to alternate between search over symbolic action sequences and optimization over configurations or paths *conditional* to such symbolic decisions. Crucial for the efficiency is how we exploit geometric information to inform the symbolic search, especially if the task objective is solely given in terms of an evaluation function $\psi(x(T))$ of the

[2] A robot with n actuated joints.

final configuration. We propose three levels of approximation for this:

- We optimize only over the final configuration $x(T)$ *conditional* to a given action sequence $a_{1:T}$. What 'conditional' really means requires to understand the 'effective end space', which we discuss in detail in the next section. This optimization over $x(T)$ is very fast and suitable as a search heuristic.
- We optimize jointly over all switch configurations $\{x(t) : t = t_1, t_2, .., t_K, T\}$ conditional to a given action sequence $a_{1:K}$. This computes explicit geometric instantiations of the action operators and optimizes the respective configurations to account also for long-term effects, e.g., optimizes over a grasp pose at $t = t_1$ to minimize also costs that arise several time steps later due to the choice of grasp pose.
- We optimize over the full path $x : [0, T] \to \mathcal{X}$ conditional to a given action sequence $a_{1:K}$. This requires methods for trajectory optimization able to cope with kinematic switches.

Readers familiar with [Srivastava *et al.*, 2014] or [Lozano-Pérez and Kaelbling, 2014] may criticize that in this staging the geometric feasibility of an action operator (e.g., the existence of a path for a grasp) is checked only on the 3rd and most costly level, optimizing over the full path. It is true that the previous approaches use proper feasibility checking (RRTs) of sub-paths in a more integrated way to inform search over action sequences. It is not the primary focus of our approach to deal with scenarios where obstacle avoidance is the major challenge. In our demonstrations, any object is reachable with the first action—but whether it should be reached depends on $\psi(x(T))$.

4.1 Level 1: Optimization Over the Effective End Space

We define the effective end space $\mathcal{X}^*(a_{1:K}) \subseteq \mathcal{X}$ as the set of all configurations $x(T)$ that can be reached with a given action sequence $a_{1:K}$, informally:

$$\mathcal{X}^*(a_{1:K}) = \{x(T) \in \mathcal{X} \,|\, a_{1:K}\} \,. \tag{7}$$

We discuss in the following more precisely what this conditioning on the action sequence means, by giving an example, discussing $\mathcal{X}^*$ as the projection of action skeleton parameters onto the end state, and introducing our specific construction of the space.

Consider a 2-step action sequence where a_1 picks up an object A from a table, and a_2 places it back on the *same* table B. In typical symbolic representations, the states before and after $a_{1:2}$ are the same; while geometrically the object may have been placed at another location on the table. In the skeleton perspective, the new object location is a parameter of the action skeleton, specifically of a_2. However, we may think of the new object location also as a free parameter of the final configuration $x(T)$. We want to capture this also in the symbolic state s_K, that is, also symbolically represent that there is a significant difference between before and after the manipulation. We do so by designing rules (for a_2) that augment the state with a `movable` predicate for the manipulated object. In our example, after the sequence we have state predicates $\texttt{movable}(A)$ and $\texttt{on}(A, B)$, which together translate to a 3D (x, y, φ)-translation-rotation joint between object A and table B and additional inequality constraints that ensure stability of the object placement on the table. In that way, the symbolic state s_K fully defines the effective end space $\mathcal{X}^*(a_{1:2})$.

We discussed earlier that our problem formulation inherently requires the symbolic representation s_k to be sufficient to define the kinematics of the world configuration $x(t)$ in the interval $t \in [t_k, t_{k+1}]$ (geometric & differential constraints (6)). Now we additionally require that s_k is a sufficient representation to define the space $\mathcal{X}^*(a_{1:k})$, which we may think of the *effective* kinematics of the end configuration $x(T)$. This "effective kinematics" describes how objects in the end configuration can be "moved" by different choices of action skeleton parameters.

An illustrative example, that we will also consider in the experiments, is the construction of a physically stable tower. Consider multiple blocks of different sizes, and boards that can be placed on top of multiple blocks. To maximize stability of the final tower we should place multiple large blocks of similar height in the bottom, a board on top, ensure that the supporting blocks span a maximal support polygon, then place multiple blocks on the board, and so on. Choosing the right placement of blocks for later stability is a long-term planning problem. However, if we can project these choices into the final configuration, thinking of all objects movable subject to the structural constraints, this problem can be solved by optimizing only over the final configuration $x(T)$ in the end space $\mathcal{X}^*$. Note that the dimensionality of $\mathcal{X}^*$ is typically lower than the number of all skeleton parameters: a grasp action a_1, for instance, has 6 grasp pose parameters. But if it is followed by a place it might project to only 3 degrees of freedom (DoFs) in $\mathcal{X}^*$ (assuming the uprightness of the object is unchanged, and the resulting DoFs are the objects (x, y, φ)-coordinates on the base).

Optimization over $x(T) \in \mathcal{X}^*$ is a standard parametric constrained (non-linear) optimization problem, just as typical robot pose optimization problems, which can be efficiently solved (in our case within about 100 msecs for 40 objects) and used as a heuristic to guide symbolic search. Clearly, optimization in the end space is only an approximation to the optimization of the full path as it neglects constraints[3] and costs arising from actually moving these objects into place. The following section addresses this.

4.2 Levels 2 & 3: Optimizing Across Kinematic Switches

Optimizing trajectories across switches in the system kinematics is challenging due to long-term dependencies. For instance, assume there is a cost gradient w.r.t. an inhand-object pose at some time slice t_2. However, the pose of the inhand-object not only depends on the current robot joint angles q_{t_2}, but also on the robot joint angles q_{t_1} at the time t_1 of grasping, which determined the rigid hand-object transformation.

[3]Note that neglecting constraints, or mutexes, is one of the core sources of finding heuristics for AI planning in general.

These long-term dependencies need to be accounted for to compute a correct trajectory gradient. Methods to cope with such problems are rare.

We address this based on our k-order motion optimization framework (KOMO) [Toussaint, 2014], where the problem over a time discretized trajectory $x_{0:T}$ is of the form

$$\min_{x_{0:T}} \quad \sum_{t=0}^{T} f_t(x_{t-k:t})^\top f_t(x_{t-k:t}) \tag{8}$$

$$\text{s.t.} \quad \forall_t : \; g_t(x_{t-k:t}) \le 0 \,, \quad h_t(x_{t-k:t}) = 0 \,, \tag{9}$$

where f_t are cost terms that approximate the control costs in the interval $[t-k, t]$ as a sum-of-squares over cliques $x_{t-k:t} = (x_{t-k}, .., x_{t-1}, x_t)$, and g_t and h_t define constraints over these cliques. By choosing $k = 2$ we can penalize and constrain arbitrary features of positions, velocities and accelerations. This problem form lends to highly efficient Gauss-Newton methods (within an Augmented Lagrangian loop) exploiting the band-diagonal structure of the pseudo-Hessian that arises from the chain structure; see [Toussaint, 2014] for details.

In a direct representation, the above mentioned long-term effects of kinematic switches break the banded structure of Jacobians and band-diagonal structure of the Hessian, loosing all the nice properties of KOMO.[4] We decided to circumvent this problem by augmenting the robot manipulator with a virtual manipulation frame with a free 6DoF joint between hand and this manipulation frame, and define $x_t \in \mathbb{R}^{n+6}$. A grasp action translates to a 6D equality constraint between manipulation frame and object frame (in Eq. (5)). We further impose zero velocity constraints on the relative grasp reference frame (object-hand pose) as long as an object is inhand.

To further simplify the trajectory optimization we impose equality constraints on the placement of objects when they are manipulated for the last time: we require their pose to be equal to the one computed by level 1 optimization. Further constraints concern standard motion optimization aspects such as collision avoidance. Control costs are squared accelerations.

We use this framework for both, level 2 and 3 optimization. When only optimizing over the keyframes (level 2) we choose a path discretization that only includes the keyframes; for the full path optimization we include 20 time steps between keyframes initialize the path optimization with an interpolation of the previously optimized keyframes. The keyframe optimization also provides a faster way to prune out action sequences (if no keyframes can be found that could generate the required grasp and place poses under geometric constraints). Concerning the optimization over time points $t_1, .., t_K$: in the experiments we will require zero accelerations at all keyframes, such that the real time duration of each interval $[t_k, t_{k+1}]$ can be chosen independently and trivially after the path optimization.

4.3 Symbolic Search

In our current implementation of the overall system we only use basic Monte Carlo Tree Search (MCTS) to search over action sequences $a_{1:K}$. We put substantial effort in efficiently computing the combinatorial set of all feasible actions (unifications of action rule preconditions with the state, using CSP methods) at every node of the tree, as this crucially decides on how the computation of rollouts scales with the number of objects. Still, plain MCTS is surely is less efficient than the existing TAMP approaches mentioned previously. Our focus is on efficient engines for the above described optimization problems.

We use a rule based representation of symbolic transitions $s_k = \mathit{succ}(a_k, s_{k\text{-}1})$. In each iteration of MCTS we unroll full action sequences $a_{1:K}$ purely symbolically, then optimize $x(t) \in \mathcal{X}^*(a_{1:K})$ w.r.t. $\psi(x(T))$ (including constraints on the final configuration). For the best of all generated final configurations we perform level 2 optimization and, if still feasible, level 3 optimization and eventually return the best solution $(x, a_{1:K}, s_{1:K})$ found.

[4] Dealing carefully with the non-banded Jacobians and neglecting the respective off-diagonal blocks in the Hessian is feasible, but in our experience overly complex and not efficient.

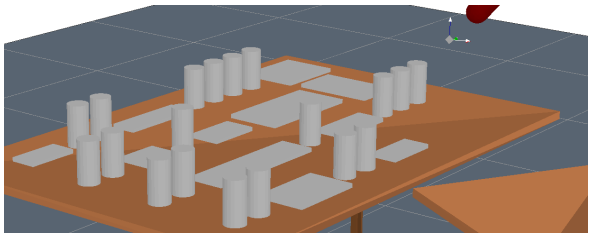

Figure 1: Typical initial configuration.

5 Experiments

We evaluated the approach on the problem of creating a stable construction from a random set of assorted boards and blocks, maximizing its height.

We defined $\psi(x(T))$ to quantify the stability and height of a construction: When a board is placed on multiple blocks, we maximize the spread of blocks subject to being within the boundary of the board, and reward the number of blocks a board is placed on. When it is placed on only one block we penalize non-centering. When blocks are placed on a board, we reward more central positionings. For brevity, we provide a detailed definition of $\psi(x(T))$ and quantitative results on achieved scores in an appendix on the author webpage.

Concerning optimization over the full manipulation path, we abstracted the robot as a 3DoF arm mounted on a floating base. We did not consider articulated fingers and optimize over finger motions for grasping as this is unrealistic to transfer to real-world. Instead we optimize the grasp pose (the relative object-hand pose), assuming that a compliant real-world gripper could perform the actual grasp. The concrete switch constraints h_{switch}, g_{switch} used in the experiment concerned non-collision (inequality), equality of grasp frame with object frame, and (in case of the last manipulation of an object) equality of the place pose with the final pose. The geometric and differential constraints h_{path}, g_{path} implement zero velocity of the object-hand pose while inhand, zero velocities and accelerations during pick and place, and non-collision. The control costs penalized accelerations and implemented a weak prior for the robot arm to be in the homing posi-

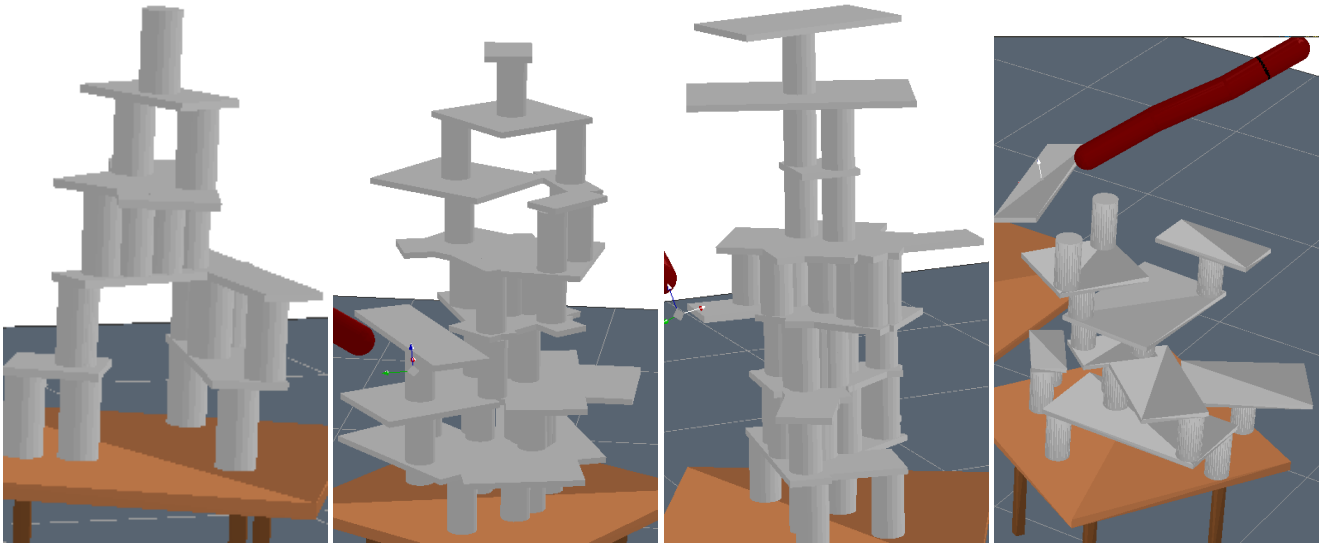

Figure 2: Samples of optimized end state configurations. Right: Snapshot of an optimized smooth full manipulation trajectory.

tion throughout (implying a useful preference to reach from above).

Figure 1 displays a typical initial configuration where the number of objects, their type and size of boards are randomized and initially placed on the grand table. They are to be assembled on the small table. Figure 2 displays typical found end configurations.

Figure 3(a) displays data on the run times on a 2.8GHz DualCore laptop for the generation of a single MCTS roll out and a single optimization over the end space configuration, depending on number of objects in random problem instances. Even for up to 100 objects—implying up to 200 pick and place manipulations and hundreds of effective DoFs in the end configuration—optimization over the end space remains below 1 second. For moderate number of objects it is from 10 to 100 msecs. Both methods are fast, effective optimization over the end space is an efficient evaluation of rollouts, and we can compute optimization configurations as those illustrated in Figure 2.

Figure 3(b) displays computation times for keyframe and full path optimizations for random problems. Note that the number of keyframes is the number of manipulations, roughly equal to twice the number of objects. Further, the paths have 20 time steps per manipulation; for 25 objects this is a (15-dimensional) trajectory with 1000 time steps across 50 manipulations. Concerning keyframe optimization we can see some outliers with more than 30-40 seconds—these are exactly cases of infeasible manipulations where we have to discard the action sequence. The resulting trajectories are smooth and collision free (if keyframe optimization indicated feasibility) and generate the optimized end state. Figure 2-right shows an example of the manipulator in action in some later phase of the assembly.

The example demonstrates success on our construction problems, leading to (locally, approximately) optimal full manipulation paths across up to 50 manipulations. We are not

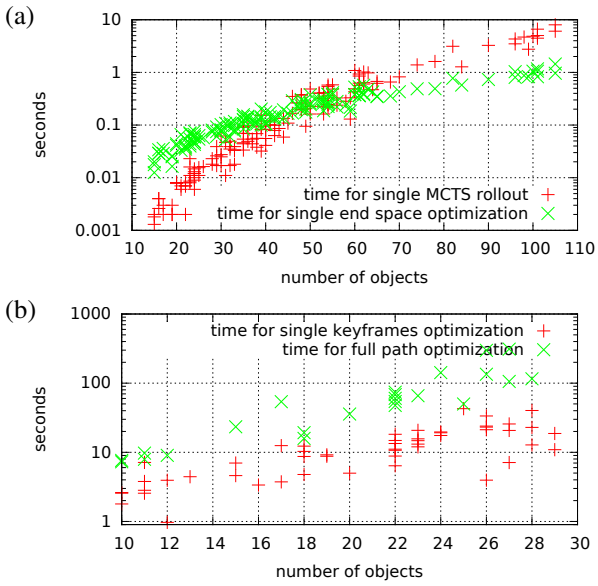

Figure 3: Running times of (a) a single MCTS rollout and a single end space optimization and (b) a keyframe optimization and full path optimization.

aware of existing methods that could address such problems in this form.

There is no doubt that existing TAMP approaches could solve an analogous feasibility problem *if* the objective was instead encoded in a symbolic way. But this demonstration shows that such problems can be solved also when the objective is given in terms of an objective function $\psi(x(T))$, and that formulating the problem as an integrated optimization problem allows us to leverage optimization methods to find (locally, approximately) optimal solutions.

6 Conclusion & Outlook

We proposed a new, optimization-based approach to sequential manipulation planning. Efficient optimization over the vast space of all possible manipulations crucially requires heuristics to inform search over the symbolic aspects. We proposed three levels for this: optimization over the end space only, over the grid of switch configurations, and over the full path. The first two levels neglect costs and constraints arising from the full path, making them faster heuristics to inform search. Other than previous TAMP methods for robot manipulation the approach can handle problems that lack a symbolic goal description and aims to provide optimal solutions instead of only feasible.

More generally, we believe that fundamental research on bridging between mathematical programming and first-order logic representations seems very fruitful. Choobineh [1992] proposed this, and Borkar et al. [2002] inversely proposed reductions of (higher-order) logical inference to *extensions* of mathematical programming (these extensions being similar to relational mathematical programs). It would be fascinating to see whether potential future solvers of general relational mathematical programs can be a basis for efficient manipulation planning methods in robotics.

Acknowledgements

The formulation of a logic geometric program was substantially influenced by discussions with K. Kersting, L. De Raedt, K. Tuyls, P. De Causmaecker, and W. Meert. The movitation towards integrated views on logic, geometric and probabilistic reasoning was driven by the inspiring work of and discussions with T. Lozano-Pérez and L.P. Kaelbling. This work was supported by the 3rdHand EU-Project FP7-ICT-2013-10610878.

References

[Alili *et al.*, 2010] Samir Alili, A. Kumar Pandey, E. Akin Sisbot, and Rachid Alami. Interleaving symbolic and geometric reasoning for a robotic assistant. In *ICAPS Workshop on Combining Action and Motion Planning*, 2010.

[Borkar *et al.*, 2002] Vivek S. Borkar, Vijay Chandru, and Sanjoy K. Mitter. Mathematical programming embeddings of logic. *Journal of Automated Reasoning*, 29(1):91–106, 2002.

[Choobineh, 1992] Joobin Choobineh. A relational model for the representation of mathematical programming models. In *System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on*, volume 3, pages 384–394. IEEE, 1992.

[de Silva *et al.*, 2013] Lavindra de Silva, Amit Kumar Pandey, Mamoun Gharbi, and Rachid Alami. Towards combining HTN planning and geometric task planning. *arXiv preprint arXiv:1307.1482*, 2013.

[Garrett *et al.*, 2014] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. FFRob: An efficient heuristic for task and motion planning. In *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2014.

[Hofmann and Williams, 2006] Andreas Hofmann and Brian Williams. Exploiting spatial and temporal flexibility for plan execution of hybrid, under-actuated systems. In *AAAI 2006*, 2006.

[Ivankovic *et al.*, 2014] Franc Ivankovic, Patrik Haslum, Sylvie Thiébaux, Vikas Shivashankar, and Dana S. Nau. Optimal planning with global numerical state constraints. In *Proceedings of 24th Int. Conf. on Aut. Planning and Scheduling, ICAPS*, 2014.

[Lagriffoul *et al.*, 2012] Fabien Lagriffoul, Dimitar Dimitrov, Alessandro Saffiotti, and Lars Karlsson. Constraint propagation on interval bounds for dealing with geometric backtracking. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 957–964. IEEE, 2012.

[Lagriffoul *et al.*, 2014] Fabien Lagriffoul, Dimitar Dimitrov, Julien Bidot, Alessandro Saffiotti, and Lars Karlsson. Efficiently combining task and motion planning using geometric constraints. *The International Journal of Robotics Research*, 2014.

[Li and Williams, 2008] Hui X. Li and Brian C. Williams. Generative planning for hybrid systems based on flow tubes. In *ICAPS*, pages 206–213, 2008.

[Lozano-Pérez and Kaelbling, 2014] Tomás Lozano-Pérez and Leslie Pack Kaelbling. A constraint-based method for solving sequential manipulation planning problems. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 3684–3691. IEEE, 2014.

[Mordatch *et al.*, 2012] Igor Mordatch, Zoran Popović, and Emanuel Todorov. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 137–144. Eurographics Association, 2012.

[Pandey *et al.*, 2012] Amit Kumar Pandey, J.-P. 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 *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on*, pages 1371–1376. IEEE, 2012.

[Plaku and Hager, 2010] Erion Plaku and Gregory D. Hager. Sampling-based motion and symbolic action planning with geometric and differential constraints. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5002–5008. IEEE, 2010.

[Posa *et al.*, 2014] M. Posa, C. Cantu, and R. Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, January 2014.

[Siméon *et al.*, 2004] Thierry Siméon, Laumond Jean-Paul, Juan Cortés, and Anis Sahbani. Manipulation planning with probabilistic roadmaps. *The International Journal of Robotics Research*, 23(7-8):729–746, August 2004.

[Srivastava *et al.*, 2014] 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 *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 639–646. IEEE, 2014.

[Sucan and Kavraki, 2011] Ioan Alexandru Sucan and Lydia E. Kavraki. Mobile manipulation: Encoding motion planning options using task motion multigraphs. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5492–5498. IEEE, 2011.

[Toussaint, 2014] Marc Toussaint. KOMO: Newton methods for k-order markov constrained motion problems. e-Print arXiv:1407.0414, 2014.

A Specifics of $\psi(x(T))$ for the tower problem

On my webpage you may find the cpp file
`source-code/15-LGP-endStateOptim.cpp`
that defines the $\psi(x(T))$ function used in this paper—this is only for reference. If you want a (non-cleaned) copy of the code snapshot that generated the experiments, contact me. (The code is far from being a library for users.)

In the tower problem, whether a stack of objects is physically stable depends on how objects support each other. In our symbolic representation s_K of the final configuration we include predicates `(support X Y)` whenever object X supports object Y (from below). Our design of the objective function $\psi(x(T))$ mainly investigates these supports. A precise quantification of stability would consider the forces exerted by the supports. In contrast, here we define easily differentiable heuristics: If a board is supported by multiple blocks we maximize their spread, subject to being within the boundary of the board. We reward if a board is placed on more blocks and penalize non-centering of supports.

The $\psi(x(T))$ in fact defines a *constrained* mathematical program in the following convention: ψ is a vector valued function, where every component of ψ either defines (i) a sum-of-squares term, (ii) an equality constraint, or (iii) an inequality constraint. The NLP is then

$$\min_x \sum_{i:t_i=\text{sos}} \psi_i(x)^2 \quad \text{s.t.} \tag{1}$$

$$\forall_{i:t_i=\text{ineq}}\ \psi_i(x) \le 0\ , \quad \forall_{i:t_i=\text{eq}}\ \psi_i(x) = 0\ , \tag{2}$$

where t_i indicates the term's type. Following the order of the source code, the terms included in $\psi(x(T))$ are:

- For every predicate `(supports X Y)` we add 4 inequality constraints to ψ, which represent the fact that object Y should be within the support area of object X. We first compute the "range" of allowed xy-coordinates of Y relative to X (which depends on the size of X), then add two inequalities to the problem for each dimension. The inequalities are scaled by a factor 10 (which should not influence the optimum, but the convergence behavior).
- For every object X that has $n \ge 2$ supporters we add n sum-of-square terms to ψ, which represent the distance of each supporter to the supporters' center, where n is the number of supporters. We first compute all locations of all supporters, and the center as their mean location. As we want to maximize the distance (the spread of supporters) we then add the sum-of-square term $0.3*(1-d)$, where d is the supporter's distance to the center in meters (and always < 1).
- For every object X that has $n \ge 1$ supporters we add three sum-of-square terms to ψ, which is equal to the 3D distance d between the supporters' center and the center of X.
- We add exactly the same terms as described by the two previous bullet points for all objects *above* every object X.

The above describes the objectives in the geometric NLP for a given symbolic final configuration s_K. To select between symbolic decisions we additionally reward $r = 10h + S$, where h is the height of the highest piece, and S counts how many supporters each object has: For each object X we add $0.2n^2$ to S, where n is the number of supporters of X. The selection criterion between tower structures is then $r - f$, where $f = \sum_{i:t_i=\text{sos}} \psi_i^2$ are the sum-of-square costs of the NLP.

Although absolute numbers have little semantics, the typical costs f for optimal(!) towers range around $[.1, .8]$, but increases drastically for non-optimized geometries or infeasible configurations. The height and support reward r is in the order of 10 or higher, depending on the number of objects.