Conflict-Directed Diverse Planning for Logic-Geometric Programming

Joaquim Ortiz-Haro[1], **Erez Karpas**[2], **Marc Toussaint**[1], **Michael Katz**[3]
[1]TU Berlin,
[2]Technion — Israel Institute of Technology,
[3]IBM Research, Yorktown Heights, NY, USA
{joaquim.ortizdeharo,toussaint}@tu-berlin.de, karpase@technion.ac.il, Michael.Katz1@ibm.com

Abstract

Robots operating in the real world must combine task planning for reasoning about what to do with motion planning for reasoning about how to do it – this is known as task and motion planning. One promising approach for task and motion planning is Logic Geometric Programming (LGP) which integrates a logical layer and a geometric layer in an optimization formulation. The logical layer describes feasible high-level actions at an abstract symbolic level, while the geometric layer uses continuous optimization methods to reason about motion trajectories with geometric constraints. In this paper we propose a new approach for solving task and motion planning problems in the LGP formulation, that leverages state-of-the-art diverse planning at the logical layer to explore the space of feasible logical plans, and minimizes the number of optimization problems to be solved on the continuous geometric layer. To this end, geometric infeasibility is fed back into planning by identifying prefix conflicts and incorporating this back into the planner through a novel multi-prefix forbidding compilation. We further leverage diverse planning with a new novelty criteria for selecting candidate plans based on the prefix novelty, and a metareasoning approach which attempts to extract only useful conflicts by leveraging the information that is gathered in the course of solving the given problem.

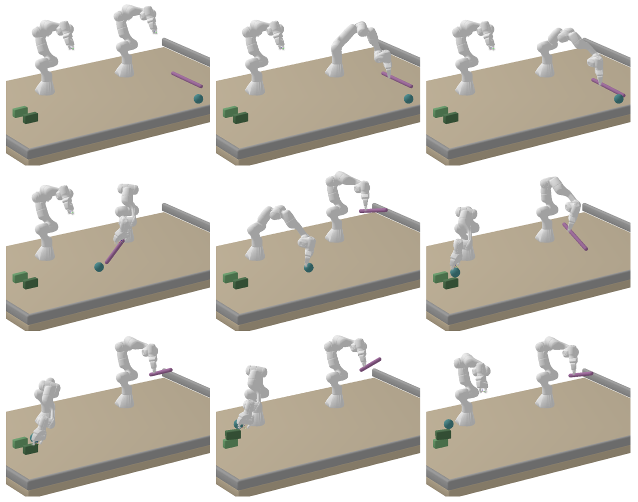

Figure 1: Example of a manipulation task solved by our algorithm. The objective is to compute high-level actions and a continuous motion that achieves the symbolic goal *(on block1 ball) (on block2 block1)* (bottom right) from the initial configuration (top left). This requires combined logic and geometric reasoning about tool-use, pushing and pick & place actions with two robot manipulators.

Introduction

Robots operating in the real world, especially robotic manipulators, must come up with plans that involve trajectories for motion. However, motion planning by itself does not reason about the logical aspects of the task at hand (LaValle 2006). Combined task and motion planning (Garrett et al. 2021) combines both symbolic planning about the task with trajectory planning for the motions involved with the plan. Many approaches to combined task and motion planning have been developed (Srivastava et al. 2014; Bidot et al. 2017; Ferrer-Mestres, Francès, and Geffner 2017; Garrett, Lozano-Pérez, and Kaelbling 2020; Lagriffoul et al. 2014) over the years.

In this paper we specifically consider Logic-Geometric Programming (LGP) (Toussaint 2015; Toussaint et al. 2018) as a formalization of combined task and motion planning which integrates logical planning into an optimization formulation of geometric reasoning and trajectory optimization. To solve an LGP problem we need to combine high-level search over the logical actions with solving the geometric and kinematic constraints that a given logical plan imposes on the underlying path optimization problem.

However, logical plans are only a necessary condition for a solution, but not a sufficient condition for geometric and kinematic feasibility. For a given symbolic plan, the geometric solver will often discover that there is no feasible motion to execute the plan, i.e. the plan is geometrically infeasible. The information about geometric infeasibility needs to be incorporated back into the logic level to propose alternative symbolic plans. A key to efficiency of solving LGPs is to reduce the number of geometric optimization problems that are checked for feasibility at the geometric level.

In this paper, we propose a new interface between symbolic planners and trajectory optimization via the identification of prefixes of logical actions that are geometrically infeasible. A key contribution to this end is a new type of compilation, called the *multi-prefix forbidding* compilation, which can also be of independent interest in diverse planning. This interface enables a systematic interaction between state-of-the-art symbolic planning and non-linear optimization methods to efficiently solve LGPs.

Based on this framework we further propose a novel extension to leverage diverse planning, based on a prefix-based novelty criteria, and a metareasoning scheme which better guides the conflict extraction of infeasible prefixes.

Background

We begin by surveying the necessary background.

Classical Planning

The logical component of an LGP can be described as a classical planning problem encoded in SAS$^+$ (Bäckström and Nebel 1995). A SAS$^+$ planning task is described by a tuple $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, g, \mathit{cost} \rangle$, where $\mathcal{V}$ is a set of *state variables* and $\mathcal{A}$ is a finite set of *actions*. Each state variable $v \in \mathcal{V}$ has a finite domain $\mathcal{D}(v)$. A *fact* is a pair $\langle v, \vartheta \rangle$ of variable $v \in \mathcal{V}$ and its value $\vartheta \in \mathcal{D}(v)$. By F we denote the set of all facts. A (partial) assignment to the variables $\mathcal{V}$ is called a *(partial) state*. We view a partial state p as a set of facts with $\langle v, \vartheta \rangle \in p$ if and only if $p[v] = \vartheta$. For a partial state p, $\mathcal{V}(p) \subseteq \mathcal{V}$ denotes the subset of state variables instantiated by p. s_0 is the *initial state*, and the partial state g is the *goal*. Each *action* a is a pair $\langle \mathit{pre}(a), \mathit{eff}(a) \rangle$ of partial states called *preconditions* and *effects*, and has an associated cost $\mathit{cost}(a) \in \mathbb{R}^{0+}$. An action a is applicable in a state s if and only if $\mathit{pre}(a) \subseteq s$. The set of all applicable actions in s is denoted by $\mathcal{A}(s)$. Applying a changes the value of $v \in \mathcal{V}(\mathit{eff}(a))$ to $\mathit{eff}(a)[v]$. The resulting state is denoted by $s[\![a]\!]$. An action sequence $\pi = \langle a_1 \dots a_K \rangle$ is a valid plan if each action is applicable in the previous state ($s_k = s_{k-1}[\![a_k]\!]$, starting from s_0), and the final state satisfies the goal, that is $g \subseteq s_K$. The set of all plans of the planning task Π is denoted by $\mathcal{P}_\Pi$. Given a sequence of actions π and a natural number k, we denote the prefix of π of length k by $\pi|_k = \langle a_1 \dots a_k \rangle$.

Logic Geometric Program

A Logic-Geometric Program (LGP) (1) is an optimization problem over both, logical decision variables and continuous decision variables, where a) logical variables $a_{1:K}, s_{1:K}$ correspond to discrete actions and states in a sequential decision domain and b) continuous variables $x(t) : t \in \mathbb{R} \to \mathbb{R}^n$ correspond to a trajectory in a continuous space.

The continuous space $\mathbb{R}^n$ represents the configuration of movable rigid objects and articulated joints, with initial condition $x_0 \in \mathbb{R}^n$. The logical variables s_k, a_k are, respectively, the symbolic state and action at step k, and can be defined with a SAS$^+$ planning task $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, g, \mathit{cost} \rangle$ (see above). $a_{1:K}$ is a short notation for $\langle a_1 \dots a_K \rangle$ (same for $s_{1:K}$). We define the successor function $\mathrm{succ}(s, a) = s[\![a]\!]$. The continuous motion is divided into K phases of fixed duration $T \in \mathbb{R}$, where nonlinear constraints $h_k(x, s_{0:k}) : \mathbb{R}^n \to \mathbb{R}^j$ (including equality constraints that can be rewritten as $h \leq \mathbf{0}$ and $-h \leq \mathbf{0}$) and cost $c(x, s_{0:k}) : \mathbb{R}^n \to \mathbb{R}$ are defined by the history of symbolic states up to k. Cost represents control effort (e.g. energy, acceleration, force and torque) and nonlinear constraints model, for example, collision avoidance, grasping, placement, and kinematic constraints.

$$\textbf{LGP} \min_{x, s_{1:K}, a_{1:K}, K} \sum_{k=0}^{K-1} \int_{kT}^{(k+1)T} c(x(t), s_{0:k})\, dt \tag{1a}$$

$$\text{s.t.} \quad x(0) = x_0, \tag{1b}$$

$$\forall k \in 0, \dots, K :$$

$$h_k(x(t), s_{0:k}) \leq 0,\ t \in [kT, (k+1)T], \tag{1c}$$

$$a_k \in \mathcal{A}(s_{k-1}), \tag{1d}$$

$$s_k = \mathrm{succ}(s_{k-1}, a_k), \tag{1e}$$

$$g \subseteq s_K. \tag{1f}$$

A solution to an LGP is a sequence of symbolic actions $a_{1:K}$ ($K \in \mathbb{N}$ is also subject to optimization) that achieves a symbolic goal g and a corresponding continuous path in configuration space $x(t)$ that fulfils the constraints $h(\cdot)$ and minimizes the cost $c(\cdot)$ defined by the action sequence. For a fixed action sequence $a_{1:K}$ the functions $c(\cdot), h(\cdot)$ are continuous and piecewise differentiable, and the optimization of the trajectory is a nonlinear program (NLP) (2), that we denote with $\textbf{NLP}(a_{1:K})$.

$$\textbf{NLP}(a_{1:K}) : \quad \min_x \sum_{k=0}^{K-1} \int_{kT}^{(k+1)T} c(x(t), s_{0:k})\, dt \tag{2a}$$

$$\text{s.t. } x(0) = x_0, \tag{2b}$$

$$\forall k \in 0, \dots, K : \tag{2c}$$

$$h_k(x(t), s_{0:k}) \leq 0,\ t \in [kT, (k+1)T].$$

where $s_{1:K}$ is uniquely defined by $a_{1:K}$ and s_0. A sequence of actions $a_{1:K}$ is geometrically infeasible when $\textbf{NLP}(a_{1:K})$ is infeasible, i.e

$$\nexists\, x(t),\ t \in [0, TK] \text{ s.t. Eq. (2b) (2c).} \tag{3}$$

Multi-Bound Tree Search

The discrete variables of the LGP formulation in Eq. (1) induce a search tree that contains sequences of symbolic states, starting from s_0. The leaf nodes where $g \subseteq s$ are potential candidates for a solution. Each node can be tested for feasibility by solving the NLP induced by the sequence of states from the root to the current node. In the standard solver (Toussaint and Lopes 2017), the tree is explored in a *breadth first seach* (BFS) order.

However, solving the continuous path problem is expensive and the number of candidate NLPs is generally too high. To alleviate this issue, Multi-Bound Tree Search (MBTS) (Toussaint and Lopes 2017) first solves relaxed versions of Eq. (2). The feasibility of each relaxed problem is a necessary condition for the feasibility of the original NLP, i.e., acting as lower bounds, whilst being computationally faster. Specifically, two bounds that consider only a subset of variables and constraints of the full path problem (2) were defined:

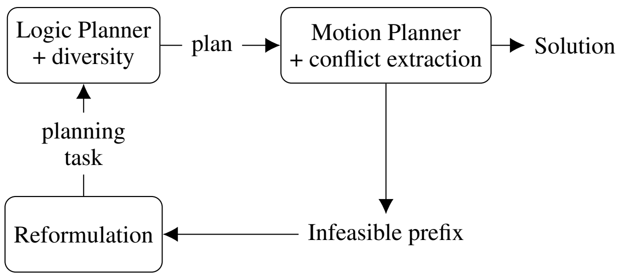

Figure 2: Overview of our approach. We combine a symbolic planner that generates plans of actions, with a motion planner to compute a trajectory. If a plan is not geometrically feasible, we extract a conflict, namely a prefix of infeasible actions, and reformulate the planning task.

The **pose bound** is a set of $k = 1, ..., K$ independent optimization problems, where problem k corresponds to the configuration at the end of phase k and has variables $x(t = kT)$, that are optimized independently.

The **sequence bound** is an optimization problem with variables $\{x(t = kT),\ \forall k = 1, ..., K\}$, that are optimized jointly accounting for their interdependencies, but without considering the continuous path between them, i.e. we only evaluate the constraints at the beginning and end of each phase. This sequence of discrete configurations is usually called keyframes or mode-switches.

Iterative Logical Planning for LGP

We can now describe our approach for solving LGP, starting with a high level description, which will be followed by more details about each of our contributions. The flowchart in Fig. 2 provides a graphical description, Algorithm 1, which appears later, shows the pseudo-code for our approach, and Fig. 3 provides an example of the execution in a simplified setting.

Our fundamental contribution in this paper is to use diverse planning on the logical component of an LGP problem, yielding a diverse sequence of logical plans. Specifically, we build upon and extend the iterative plan forbidding approach (Katz and Sohrabi 2020) to generate this sequence of logical plans. Each logical plan is then checked for geometric feasibility by calling a motion planner (we use the trajectory optimization algorithm from (Toussaint 2015)).

While even the basic scheme proposed above can outperform the baseline planning algorithms of Multi Bound Tree Search (MBTS), we propose several improvements. First, we exploit the fact that if the geometric feasibility checking for a logical plan π failed, we can often extract a *conflict*, which is similar to conflicts in conflict-directed clause learning (Silva and Sakallah 1999) or conflict-directed A* (Williams and Ragno 2007). In this paper, we address conflicts in the form of plan *prefixes* – that is, a sequence of logical actions π which is applicable from the initial state at the logical level, but has no feasible geometric trajectory.

To exploit such conflicts, we must do two things. First, we must be able to efficiently extract a conflict from a logical plan which is not geometrically feasible. Second, we must forbid our logical planner from generating plans which contain the identified conflict as a prefix. Both of these will be discussed in detail later.

One strength of our algorithm is that it makes very little assumptions (namely, that the problem can be formulated as an LGP, which is very general and applicable to any problem in robotic sequential manipulation with complex physics interactions) and can be implemented based on two off-the-shelf components.

- A motion optimization framework that formulates **NLP**$(a_{1:K})$ and provides a binary output $\{0, 1\}$ (infeasible or feasible with the corresponding trajectory).
- A logic planner to solve a discrete planning problem.

Prefixes as Conflicts

To begin the detailed discussion of our approach, we discuss why we choose to identify prefixes as conflicts, and not a more general restriction on plans. The main reason for this is that the logical planning task is merely an abstraction of the LGP task, which completely ignores the geometric aspects of the problem.

Theorem 1. *Let* Π *be an LGP task* (1)*, and let* π *be a sequence of logical actions, such that* π *is not geometrically feasible from the initial state. Then any sequence of actions* π' *which contains* π *as a prefix is not geometrically feasible from the initial state.*

Proof sketch. Recall that a sequence $\pi = \langle a_1 \dots a_K \rangle$ of K actions is not geometrically feasible if the nonlinear optimization problem **NLP**(π) (2) is infeasible, Given a plan $\pi' = \langle \pi, a'_1 \dots a'_J \rangle$ that contains π as prefix, assume there exists a geometric path $x(t),\ t \in [0, (K + J)T]$ that is feasible for **NLP**$(\langle \pi, a'_{1:J} \rangle)$. This implies that $x(t),\ t \in [0, KT]$ is also feasible, which contradicts that **NLP**(π) is infeasible, because variables and constraints in time interval $t \in [0, KT]$ are the same in **NLP**$(\langle \pi, a'_{1:J} \rangle)$ and **NLP**(π). □

For example, consider an LGP task in which the robot can pick and place objects on a cluttered table. Suppose the starting sequence $\langle \text{pick}(B) \rangle$ is feasible at the geometric level, but $\langle \text{pick}(B), \text{place}(B) \rangle$ is infeasible. Then it is safe to infer that any action sequence beginning with $\langle \text{pick}(B), \text{place}(B) \rangle$ will be infeasible, as demonstrated in the previous theorem.

However, it is *not* safe to infer that $\langle \text{pick}(B), \text{place}(B) \rangle$ can never be applied. For example, it is possible that object A is obstructing the place of object B, and thus the sequence of actions $\langle \text{pick}(A), \text{place}(A), \text{pick}(B), \text{place}(B) \rangle$ might be geometrically feasible. Note that in this example, pick(A) and place(B) form a causal link (Tate 1977), as pick(A) supports place(B). However, this causal link does not appear at the logical level, but only at the geometric level. It is not possible to infer a stronger conflict than prefixes without a deeper analysis of the geometric feasibility – this is beyond the scope of this paper, but will be explored in future work.

We remark that prefix forbidding is a general and sound way to encode information from the geometric level back into the symbolic level. It does not rely on hand-crafted additional predicates or checks, and is therefore applicable to

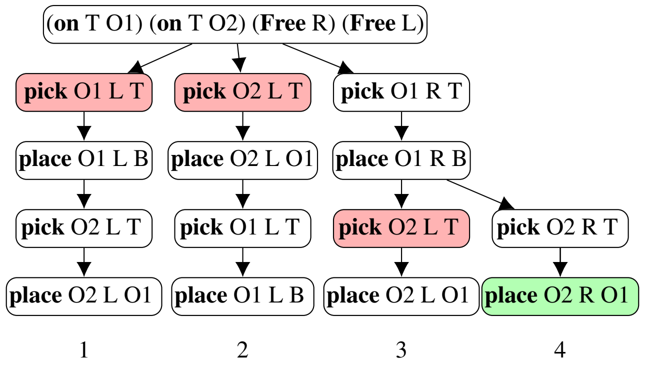

Figure 3: Illustrative example of the execution of our algorithm (with *N=1* and *eager* conflict extraction). The scene contains two movable objects *O1*, *O2*, a table *T*, a box *B* and two robots *R*, *L* that can *pick* and *place* the objects. The goal is *(on B O1)*, *(on O1 O2)*. In each iteration, the planner has produced one plan (1,2,3,4, in this order) that has been tested for feasibility. The motion planner returned the minimal prefix of infeasible actions (highlighted in red), that is used to reformulate the planning task for the next iterations. Plan number 4 (in green) is geometrically feasible.

any sequence of actions, independently of the underlying physics or geometry model.

Forbidding Plans by Prefixes

Having explained why prefixes are an important part of this approach, we now describe how we prevent our planner from returning logical plans which begin with a given set of prefixes. Our approach builds upon previous work (Katz et al. 2018), which has suggested a *plan forbidding reformulation*, a way of constructing a planning task with a set of valid plans being reduced by precisely the given plan. The suggested construction follows the execution of the given plan (sequence of actions) $\langle a_1 \dots a_K \rangle$ and allows to achieve the (modified) goal only once an action different from a_k was applied at step k.

We modify this construction in two ways. First, instead of forbidding $\langle a_1 \dots a_K \rangle$ as a plan, we forbid it as a prefix, so that applying the starting sequence $\langle a_1 \dots a_K \rangle$ in the reformulated task leads to a dead end. Thus, there are no plans for the reformulation with the prefix $\langle a_1 \dots a_K \rangle$.

Second, we simultaneously forbid multiple prefixes. While this effect can be achieved by sequential application of forbidding a single prefix, the simultaneous forbidding approach yields a much more compact compilation. The key to the simultaneous forbidding approach is building a *prefix tree* which contains all (non-dominated) prefixes. A prefix $\tilde{\pi}$ is dominated by prefix π if π is a prefix of $\tilde{\pi}$, in which case it is enough to forbid π and not $\tilde{\pi}$. We construct a tree $T = (N, E)$ where each node corresponds to a prefix, and there is an edge from node π to node π' if we can add one action to π to yield π'. Given a set of prefixes, this tree can be efficiently constructed by adding the nodes from each prefix iteratively. Given a prefix tree, Definition 1 shows how to construct a planning task that forbids exactly these prefixes.

Definition 1. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, g, cost \rangle$ be a planning task, $T = (N, E)$ be a prefix tree with $L \subseteq N$ being the leaf nodes, and $\mathcal{A}(T)$ be the set of operators that appear on the prefixes in T. The task $\Pi_T^- = \langle \mathcal{V}', \mathcal{A}', s_0', g', cost' \rangle$ is defined as follows.*

- *$\mathcal{V}' = \mathcal{V} \cup \{\overline{v}\} \cup \{\overline{v}_s \mid s \in N\}$, with $\overline{v}_s$ being binary variables and $\overline{v}$ being a ternary variable,*
- *$\mathcal{A}' = \mathcal{A}^e \cup \mathcal{A}^1 \cup \mathcal{A}^2 \cup \mathcal{A}^3$, where $\mathcal{A}^e = \{a^e \mid a \in \mathcal{A} \setminus \mathcal{A}(T)\}$, $\mathcal{A}^1 = \{a^1 \mid a \in \mathcal{A}\}$, $\mathcal{A}^2 = \{a^2 \mid a \in \mathcal{A}(T)\}$, and $\mathcal{A}^3 = \{a^3_{(s,t)} \mid (s,t) \in E\}$ with*
$$a^e = \langle pre(a) \cup \{\langle \overline{v}, 1\rangle\}, \mathit{eff}(a) \cup \{\langle \overline{v}, 0\rangle\}\rangle$$
$$a^1 = \langle pre(a) \cup \{\langle \overline{v}, 0\rangle\}, \mathit{eff}(a)\rangle$$
$$a^2 = \langle pre(a) \cup \{\langle \overline{v}, 1\rangle\} \cup \{\langle \overline{v}_s, 0\rangle \mid (s,t) \in E^a\}, \mathit{eff}(a) \cup \{\langle \overline{v}, 0\rangle\}\rangle$$
$$a^3_{(s,t)} = \langle pre(a_{(s,t)}) \cup \{\langle \overline{v}, 1\rangle, \langle \overline{v}_s, 1\rangle\}, \mathit{eff}(a_{(s,t)}) \cup \{\langle \overline{v}_s, 0\rangle, \langle \overline{v}_t, 1\rangle\}\rangle \text{ if } t \notin L,$$
$$a^3_{(s,t)} = \langle pre(a_{(s,t)}) \cup \{\langle \overline{v}, 1\rangle, \langle \overline{v}_s, 1\rangle\}, \{\langle \overline{v}, 2\rangle\}\rangle \text{ if } t \in L,$$
$$cost'(a^e) = cost'(a^1) = cost'(a^2) = cost'(a^3) = cost(a),$$
- *$s_0'[v] = s_0[v]$ for all $v \in \mathcal{V}$, $s_0'[\overline{v}] = 1$, $s_0'[\overline{v}_{s_0}] = 1$, and $s_0'[\overline{v}_s] = 0$ for all $s \in N \setminus \{s_0\}$, and*
- *$g'[v] = g[v]$ for all $v \in \mathcal{V}$ s.t. $g[v]$ defined, and $g'[\overline{v}] = 0$.*

The following theorem states the correctness of the compilation.

Theorem 2. *Let Π be a planning task, let P be a set of prefixes and T be its prefix tree. If Π_T^- is the planning task described in Definition 1, then there is a bijective mapping between $\mathcal{P}_{\Pi_T^-}$ and $\{\pi \in \mathcal{P}_\Pi \mid \forall k, \pi|_k \notin P\}$.*

Proof sketch. The proof is similar to the proof of Theorem 6 by Katz et al. (2018). The main difference between the two reformulations is in the application of $a^3_{(s,t)}$ for $t \in L$, which leads to a dead end state. Therefore, for any plan π for Π with a prefix in P, a corresponding sequence of actions $r^{-1}(\pi)$ is not a plan for Π_T^-. □

Finally, we remark that the conflicts we extract can be encoded as PDDL 3 trajectory constraints (Gerevini et al. 2009). These can be compiled away (Baier and McIlraith 2006), and the above-mentioned compilation is a special case of such a compilation.

Feasibility Checking

Having described how the logical planning works, we now explain in more detail how we perform geometric feasibility checking for a candidate logical plan, and how we can extract a logical conflict in the process.

If $\pi = \langle a_1 \dots a_K \rangle$ was found to be geometrically feasible, then we have found a solution to our LGP task and can terminate. Otherwise, we can stop and return the conflict we have at hand $\langle a_1 \dots a_K \rangle$. We will refer to doing this as *lazy* conflict extraction. Alternatively, we can also search for a stronger conflict, in the form of a shorter prefix of π

that is not geometrically feasible, which we refer to as *eager* conflict extraction. *Eager* conflict extraction searches for the strongest possible conflict we can extract from π, that is, the shortest prefix $\pi|_k$ which is geometrically infeasible, i.e.

$$\min\ k \text{ s.t. } \text{Feas}(\pi|_k) = 0 \tag{4}$$

where $\text{Feas}(\pi)$ is a binary function that returns 0 (if π is infeasible) or 1 (if π is feasible). By Theorem 1, $\text{Feas}(\pi|_k) \geq \text{Feas}(\pi|_{k+1})$. Therefore, we can find the strongest conflict with a binary search for the length k of this prefix. Initially, the lower bound l is initialized to 0, and the upper bound u is initialized to K . The evaluation of the $\text{Feas}(\pi|_m)$, for the midpoint $m = \lfloor \frac{l+u}{2} \rfloor$ corresponds to checking with the motion planner whether the prefix up to m is feasible.

As a side note, we can also evaluate the pose bound before the binary search operation. We first check whether $\langle a_1 \dots a_K \rangle$ has feasible geometric poses for $k = 1 \dots K$. If the pose checking failed for some k, we can infer that $\langle a_1 \dots a_k \rangle$ is infeasible, and start binary search with $u = k$.

As the geometric feasibility checking is the most expensive computational action we perform, we cache every prefix we check and whether it is feasible or not. This cache is helpful in speeding up feasibility checking, as different logical plans might still share a common prefix. Additionally, this cache serves as a dataset which captures the history of computational actions performed so far, which will be useful for (a) metareasoning about feasibility checking, and (b) guiding our choice of which plan to check for feasibility next when we use diverse planning. These are described in the following sections.

Metareasoning for Conflict Extraction

So far we have described two approaches for conflict extraction: *lazy*, which does not perform any reasoning to extract conflicts, and *eager*, which attempts to find the minimal conflict. We now describe a middle-ground approach, which tries to perform metareasoning (Russell and Wefald 1991) to balance the cost (the computational effort spent on extracting a conflict) and the reward (the benefits from having a stronger conflict). As the most expensive operation in our algorithm is geometric feasibility checking, we measure both the reward and the cost in number of geometric feasibility checks – either required to extract the conflict, or saved by having the conflict.

We now describe the metareasoning problem we face in deciding when to stop looking for a conflict, and the overall utility we can expect to obtain. Let $\tau = \langle a_1 \dots a_k \rangle$ be some sequence of actions. We will denote by $r(\tau)$ the reward from adding the conflict τ. Of course, this is an unknown quantity, and we will describe ways to estimate it later. Recall that during the binary search for a minimal conflict, we have a logical plan π, and a range $[l, u]$ such that $\pi|_u$ is not geometrically feasible, while $\pi|_l$ is. Thus, we can define the metareasoning problem for a given plan π of length $|\pi|$ as a Markov Decision Process (MDP) (Bellman 1957) with states $S_\pi = \{\langle l, u \rangle \mid l \leq u \in 0 \dots |\pi|\}$ – that is, each state describes the current range of the search.

The terminal states are those where the search has converged, that is $\{\langle l, l \rangle \mid l \in 0 \dots |\pi|\}$. The reward in state $\langle l, l \rangle$ is the reward from adding the conflict $\pi|_l$, that is $r(\pi|_l)$. The reward from all other states is 0. As we are sure to reach a terminal state, there is no need to introduce a discount factor (that is, $\gamma = 1$).

The possible actions at state $\langle l, u \rangle$ are either to stop searching or continue searching. The decision to stop searching yields a deterministic transition to the state $\langle u, u \rangle$ – that is, we terminate and add the conflict $\pi|_u$, obtaining reward $r(\pi|_u)$.

Although the binary search always continues search by checking the middle node ($\lfloor (l + u)/2 \rfloor$), using the metareasoning MDP allows us to consider any of the nodes between l and u as the next node to check. Thus, we have $u - l + 1$ possible actions – one for each node in the range.

Let us denote the probability of a sequence of actions τ being geometrically feasible by $p_f(\tau)$. Then by continuing the search to node m (representing $\pi|_m$), we reach the state $\langle m, u \rangle$ with probability $p_f(\pi|_m)$, and state $\langle l, m \rangle$ with probability $1 - p_f(\pi|_m)$.

Due to the structure of this MDP which lacks any loops, we can compute the optimal values using simple dynamic programming, starting with the terminal states, and computing the optimal value function for states with an increasing gap between the lower and upper bound – that is, we compute the value for states $\{\langle l, l+1 \rangle \mid l \in 1 \dots |\pi| - 1\}$, then for $\{\langle l, l+2 \rangle \mid l \in 1 \dots |\pi| - 2\}$, and so on.

Of course, we still do not know the exact rewards or transition probabilities. In the following, we describe a data-driven method to estimate these, which allows us to compute an optimal policy for an approximate MDP.

Data-driven Estimation

Although computing $r(\tau)$ exactly is not tractable, it should be commensurate with the number of logical plans which would be pruned by introducing the conflict τ. While we do not know this number, we can estimate the fraction of plans that would be pruned by conflict τ by the fraction of plans we have discovered which have τ as a prefix. Thus, we can define the estimator

$$\hat{r}(\tau) := \frac{|\{\pi' \mid \pi' \in C, \tau \text{ is a prefix of } \pi'\}|}{|C|}$$

where C is the set of prefixes in our cache. As the number of matching prefixes in the numerator might be 0 (especially early on in the process), we actually add 1 to both numerator and denominator.

The probability of a prefix being feasible or not can also be estimated from the history of previous feasibility checks. Recall that we cache every prefix we check for feasibility. We use this cache to estimate p_f. We follow a type system approach (Lelis 2013) and define a set of simple features for each prefix. Specifically, we use the length of the prefix as its only feature, and keep track of how many feasibility checks were performed for each prefix length, and how many of these turned out to be feasible – the ratio between these is our estimate of p_f, denoted $\hat{p_f}$.

Combining $\hat{r}$ and $\hat{p_f}$ we can define our MDP. As our empirical results will show, this approach results in reducing the runtime of our solver.

Diverse Logical Planning for LGP

So far, we discussed an incremental approach, which generates one new logical plan at every iteration, and then tests that plan. We now explain how we can explore the space of logical plans more rapidly. The key idea here is to generate multiple plans at each iteration, and then choose one of them for geometric feasibility checking. Generating a set of plans is done by applying the forbidding compilation (Definition 1) iteratively, as done in previous diverse planning approaches (Katz et al. 2018).

The main question here is how to choose which plan to test next. Prefixes drive our approach, so it makes sense to choose a plan which has the longest novel prefix, as even if that plan fails there is a higher chance that we will extract a short conflict from that plan. Furthermore, choosing a plan with a novel prefix encourages our approach to explore the space of logical plans, thereby covering diverse high-level approaches to the task that imply different nonlinear programs for the continuous trajectory. Thus, we define the novelty of a plan π with respect to a set of plans LP as

$$np(\pi, LP) := -\min\{k \mid \forall \pi' \in LP, \pi'|_k \neq \pi|_k\}$$

We then choose to test the plan π which maximizes the novelty with respect to the set of plans that were already tested for geometric feasibility, breaking ties randomly. We remark that this notion of novelty is different from previous ones (Lipovetzky 2021; Tuisov and Katz 2021), and serves as a greedy selection criterion for choosing the next plan. It is not clear how to extend this idea into a metric which would allow choosing multiple different plans which maximize mutual novelty – this could be explored in future work.

To summarize, Algorithm 1 describes our technique in pseudo-code and Figure 3 shows an illustrative example of the execution of our algorithm in a simplified setting. We can now state the theorem proving that our approach is sound and complete.

Theorem 3. *If the underlying classical planner is sound and complete and the motion planner always finds a feasible trajectory if such a trajectory exists* (2)*, then Algorithm 1 is sound and complete.*

Proof sketch. The proof follows from the fact that we only identify prefixes which can not appear in the beginning of geometrically feasible plans (Theorem 1), and from the correctness of the forbidding compilation (Theorem 2). □

An important technical point is that some planners perform a relevance analysis, and discard actions or state variables which are thought to be useless or redundant. For example, two actions might have the same logical effects, and thus the planner might decide to keep only one of them. However, the actions might lead to different geometric constraints, and it might be the case that one of them is feasible while the other is not. Thus, such preprocessing techniques must be disabled when solving the logical planning task.

Algorithm 1: Pseudo-code for Diverse LGP Planning

```
Input: LGP task $\Pi_{\text{LGP}}$
Parameters: $N$    ▷ number of plans to generate at each iteration
$\Pi$ := logical projection of $\Pi_{\text{LGP}}$
$T := \emptyset$    ▷ set of tried plans
$LP := \emptyset$    ▷ set of found logical plans
$MC := \emptyset$    ▷ set of found conflicts
while not solved do
    $\Pi^f :=$ FORBID$(\Pi, LP \cup MC)$    ▷ Forbid found plans and conflicts. See Sec. Forbidding Plans by Prefixes
    $LP := LP \cup$ Diverse-Plan$(\Pi^f, N)$    ▷ call diverse logical planner to find $N$ new plans.
    $\pi :=$ SELECT$(LP, T)$    ▷ select a plan to try. See Sec. Diverse Logical Planning for LGP.
    feasible, traj := MOTION-Feasible?$(\Pi_{\text{LGP}}, \pi)$    ▷ Check $\pi$ for geometric feasibility
    if feasible then return $\pi$, traj    ▷ Return trajectory and logical plan
    else
        $T := T \cup \{\pi\}$
        conflict := FIND-CONFLICT$(\pi)$    ▷ find a prefix of $\pi$ that is infeasible. See Sec. Feasibility Checking and Sec. Metareasoning for Conflict Extraction
        $MC := MC \cup \{$conflict$\}$
    end if
end while
```

Related Work

We now discuss some related work about combined task and motion planning (TAMP). Perhaps the closest approach to ours is the extensible planner-independent interface layer between task and motion planners (Srivastava et al. 2014). It combines a black-box task planner with a motion planner through reformulation. It calls a task planner on an abstract task, and if it fails it encodes geometric information about the cause of failure into the task planner with special predefined predicates. Similarly, the method in (Dantam et al. 2016) incrementally incorporates information about the motion feasibility into the symbolic description by using a constraint-based task planning formulation and a Satisfiability Modulo theory (SMT) solver. Another related approach is PDDLStream (Garrett, Lozano-Pérez, and Kaelbling 2020), which also combines symbolic planners with motion planning. Specifically, PDDLStream uses constrained samplers (in configuration space) to discretize the motion planning problem, and PDDL planning to solve the combined problem. (Ferrer-Mestres, Francès, and Geffner 2017) argues that classical planners can solve task and motion planning problems through a precompilation of geometric information.

The above methods address a path finding formulation of TAMP and use sample-based path planning, rather than an optimization based formulation such as LGP that couples logic search with constrained optimization. Both formulations have strengths and weaknesses in different situations, and in this paper we focus on efficiently solving the optimization-based formulation. Further, in the above approaches, geometric information is fed back by explicitly encoding it in the logical layer, enriching the language of log-

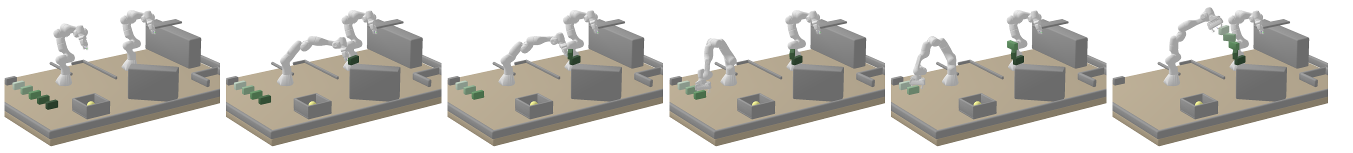
Figure 4: The goal in the *blocks* environment is to rearrange objects in the scene, for example, building a tower of 5 blocks. In total, the scene contains 12 movable objects, that can be manipulated by two robots and placed on top of each other.

ical planning with additional predicates. This is in contrast to our approach which keeps the "languages" of the two layers separate by translating geometric infeasibility constraints into forbidden prefixes of logical actions.

Empirical Evaluation

Benchmarks

We use 3 different domains, all with 2 7-DOF robotic arms.

Blocks the robots can execute pick and place actions to construct a specified tower of blocks, similarly to the classical blocksworld domain – except that the planner must come up with motion plans as well. Robots can hold a stack of blocks, move the boxes and place several objects on top of other objects. See Fig. 4.

Hanoi the robots can execute pick and place actions, to solve a tower of Hanoi problem with objects of equal size and three tables. Only the top object of each tower can be picked and at most one tower is allowed in each table. From a symbolic point of view, this is harder than *blocks* and requires longer action sequences, but the instances we use have less movable objects in the scene.

Push the robots can pick and place blocks and balls, and pick sticks and use them as a tool to push balls. The goal is to move balls and blocks to a desired symbolic state, for example, stacking blocks and placing the ball on top. See Fig. 1.

For each domain, we generate different problems (e.g *blocks*-{*0,1,2,3,4,5*}) by modifying the goal and the number of objects, increasing the complexity in the logical and geometric levels. Our benchmark contains 6 problems in the domain *blocks*, 3 in *Hanoi* and 11 in *Push*. A subset of these problems, together with the solutions computed by our framework, are shown in the supplementary video https://youtu.be/7Ev6zNbqdjo.

LGP Formulation

Logic The LGP formulation uses a minimal logical description that encodes only the structure of the kinematic tree and the type of physical interactions (whether stable grasp or pushing). Examples of predicates and actions are: *on(A B)*, *busy(gripper)* and *(pick A gripper T)*, *(push A stick table)*.

Geometry Collision avoidance, reachability, physical interactions and placement constraints are modelled in the geometric level. **Grasping model:** Instead of force-based grasp constraints we use simplified geometric constraints, namely that a point near the endeffector's grasp palm touches the object surface. In practice, for boxes this typically implies the existence of a stable grasp, which is then represented as a stable relative transformation until placement. **Pushing model:** We introduce decision variables for the force and point-of-attack between two interacting objects to model pushing. Motion and forces are then constrained by physics equations (Toussaint, Ha, and Driess 2020).

Baselines

We compare our new approach "Diverse Logical Planning" against three variations of Multibound Tree search *MBTS*-{*0,1,2*}. *MBTS-0* does not perform geometric checks in intermediate symbolic nodes, i.e. it waits until a full candidate symbolic plan is found. *MBTS-1* and *MBTS-2* check, respectively, the pose and sequence bound before expanding a symbolic node in the BFS search.

Geometric checks during node expansion (*MBTS-1*, *MBTS-2*) prune partial plans that are infeasible, which reduces the branching factor of the search and future node expansions, but increases computational time spent in solving NLPs of action sequences that do not lead to the goal.

The scope of this paper is to improve the logic search specifically in the LGP framework, so that the formulation can be applied to settings that require longer action sequences and challenging symbolic reasoning. Therefore, we do not compare to other methods in task and motion planning, e.g. (Garrett, Lozano-Pérez, and Kaelbling 2020), that use different underlying problem formulations and methods.

Results

We use the first iteration of LAMA (Richter and Westphal 2010) as our underlying classical planner. We ran a set of experiments comparing several versions of our approach to the baselines – all experiments were run on an AMD Ryzen 9 5980HS CPU with a 600 second time limit per run. Results are shown in Tab. 1. We omit problems *hanoi-2* and *blocks-5*, that were not solved by any algorithm or baseline.

Comparison to baseline Hypothesis: *"Our basic novel approach (N=1, eager conflict extraction) will be faster and solve more problems than any of the MBTS baselines"*.

Our method with "$N = 1$, *eager*" solves more problems (18 vs 12 out of 20) and is faster (16 vs 2) than all the baselines *MBTS*-{*0,1,2*}. In Table 1, we only report *MBTS-0*, that shows better performance than the other baselines.

MBTS-0 does not solve problems that require long action sequences or where the branching factor of the tree is very high (for example, the domain *blocks* contains 12 movable objects). Due to the logically-uniformed behaviour

	MBTS-0			N=1 *eager*			N=4 *eager*			N=4 *meta*		
	time	pose	seq	time	pose	seq	time	pose	seq	time	pose	seq
blocks-0	**19.4**$_{1.0}$	12.0$_{0.0}$	3.0$_{0.0}$	43.7$_{1.8}$	19.9$_{0.9}$	6.5$_{0.5}$	41.8$_{4.3}$	17.5$_{1.9}$	6.5$_{0.9}$	41.3$_{4.4}$	17.5$_{1.9}$	2.8$_{0.6}$
blocks-1	-	-	-	44.8$_{1.3}$	18.0$_{0.0}$	5.0$_{0.0}$	**44.0**$_{5.6}$	17.1$_{2.3}$	4.8$_{0.9}$	46.5$_{6.4}$	17.1$_{2.3}$	1.7$_{0.3}$
blocks-2	-	-	-	82.6$_{10.5}$	17.0$_{0.0}$	4.0$_{0.0}$	**60.4**$_{8.6}$	12.6$_{1.1}$	2.4$_{0.5}$	70.9$_{11.8}$	12.6$_{1.1}$	1.4$_{0.2}$
blocks-3	-	-	-	111$_{5.8}$	17.0$_{1.0}$	3.4$_{0.4}$	104$_{19.7}$	21.7$_{2.8}$	5.4$_{1.1}$	**80.2**$_{12.0}$	20.8$_{3.1}$	2.1$_{0.4}$
blocks-4	-	-	-	200$_{33.1}$	27.1$_{8.2}$	6.1$_{2.8}$	160$_{17.0}$	19.3$_{3.1}$	3.3$_{1.1}$	**139**$_{22.6}$	17.8$_{2.7}$	1.3$_{0.2}$
hanoi-0	10.4$_{0.4}$	13.0$_{0.0}$	4.0$_{0.0}$	**7.0**$_{0.2}$	7.0$_{0.0}$	3.0$_{0.0}$	10.0$_{1.9}$	8.0$_{0.9}$	3.7$_{0.8}$	9.1$_{1.8}$	8.6$_{1.1}$	2.9$_{0.5}$
hanoi-1	34.7$_{0.7}$	34.0$_{0.0}$	6.0$_{0.0}$	27.0$_{0.6}$	17.0$_{0.0}$	8.0$_{0.0}$	18.7$_{3.1}$	13.5$_{1.0}$	5.1$_{0.6}$	**13.8**$_{2.2}$	14.0$_{1.0}$	3.4$_{0.4}$
push-1	41.9$_{0.8}$	55.8$_{0.2}$	1.0$_{0.0}$	**17.1**$_{0.4}$	14.0$_{0.0}$	4.0$_{0.0}$	24.4$_{1.4}$	17.3$_{1.1}$	5.3$_{0.5}$	24.9$_{1.7}$	18.7$_{1.2}$	3.8$_{0.4}$
push-2	50.0$_{1.0}$	64.0$_{0.0}$	1.0$_{0.0}$	49.5$_{0.9}$	37.0$_{0.0}$	13.2$_{0.1}$	37.1$_{1.1}$	23.2$_{0.9}$	7.2$_{0.4}$	**34.3**$_{1.6}$	24.3$_{0.8}$	3.2$_{0.2}$
push-3	27.7$_{0.9}$	38.0$_{0.0}$	1.0$_{0.0}$	**14.4**$_{0.2}$	11.0$_{0.0}$	3.0$_{0.0}$	26.1$_{3.2}$	17.9$_{2.0}$	5.8$_{0.9}$	21.1$_{1.8}$	17.3$_{1.8}$	2.9$_{0.3}$
push-4	75.9$_{1.5}$	104$_{0.0}$	2.0$_{0.0}$	71.3$_{7.8}$	41.2$_{2.8}$	15.2$_{1.5}$	32.6$_{4.1}$	20.3$_{2.2}$	5.9$_{1.0}$	**30.6**$_{2.7}$	21.3$_{2.4}$	3.1$_{0.4}$
push-5	111$_{1.6}$	144$_{0.1}$	1.0$_{0.0}$	**20.4**$_{0.3}$	17.0$_{0.0}$	5.0$_{0.0}$	30.8$_{2.5}$	23.7$_{2.2}$	7.4$_{0.9}$	29.4$_{2.4}$	24.4$_{2.3}$	3.2$_{0.4}$
push-6	117$_{1.5}$	142$_{0.0}$	1.0$_{0.0}$	64.5$_{1.2}$	50.0$_{0.0}$	17.1$_{0.1}$	45.4$_{1.0}$	29.2$_{0.7}$	9.2$_{0.4}$	**45.1**$_{1.2}$	31.8$_{1.2}$	4.6$_{0.3}$
push-7	-	-	-	**68.6**$_{4.6}$	51.3$_{3.5}$	19.0$_{1.6}$	79.1$_{5.4}$	52.6$_{4.0}$	18.0$_{1.6}$	70.4$_{2.6}$	53.0$_{2.7}$	7.4$_{0.5}$
push-8	78.3$_{1.2}$	92.0$_{0.0}$	1.0$_{0.0}$	**17.0**$_{0.4}$	13.0$_{0.0}$	3.0$_{0.0}$	32.4$_{3.6}$	26.0$_{3.0}$	7.8$_{1.1}$	32.8$_{3.7}$	28.2$_{3.6}$	4.1$_{0.6}$
push-9	248$_{40.1}$	423$_{67.2}$	2.5$_{0.5}$	63.3$_{1.3}$	45.0$_{0.0}$	16.0$_{0.0}$	**46.2**$_{6.2}$	32.5$_{3.7}$	10.4$_{1.4}$	49.8$_{11.9}$	39.7$_{9.5}$	5.3$_{1.7}$
push-10	**12.7**$_{0.5}$	16.0$_{0.0}$	1.0$_{0.0}$	12.8$_{0.5}$	9.0$_{0.0}$	3.0$_{0.0}$	13.8$_{1.6}$	10.4$_{1.3}$	3.3$_{0.6}$	13.2$_{1.4}$	10.5$_{1.3}$	1.6$_{0.2}$
push-11	-	-	-	61.1$_{9.3}$	25.5$_{2.3}$	10.7$_{1.4}$	**26.0**$_{2.0}$	13.4$_{0.5}$	3.8$_{0.4}$	30.5$_{5.4}$	16.7$_{2.5}$	2.6$_{0.6}$
Total	827	1138	24.5	976	437	145	833	376	115	783	394	57.4

Table 1: Summary of the experimental results. We report the computational time in seconds (*time*), and the number of calls to the motion planner for checking the pose bound (*pose*) and the sequence bound (*seq*), with the mean over 10 randomized runs in black and standard deviation of the mean estimator in gray. Total is the sum of the columns (note that the sum for *MBTS-0* is over less problems). A dash "-" denotes that the problem was not solved in all 10 runs.

of Breadth First Search, it only finds few symbolic plans (sometimes zero), none of them geometrically feasible. Instead, our method leverages state of the art symbolic planning to compute action sequences efficiently even in huge symbolic spaces, and geometric information is encoded incrementally in the planning task through our prefix forbidding reformulation.

Analysis of Diverse Planning Hypothesis: *"Diverse planning with novelty measure will improve over incremental plan generation"*.

We compare "*N=1, eager*" (the planner produces a single plan, that is evaluated by the motion planner) and "*N=4, eager* " (the planner produces 4 plans in each iteration, that are stored in a buffer; the motion planner evaluates the plan in the buffer that maximizes our novelty criteria).

$N = 4$ reduces both the overall computational time and the number of tested plans. Choosing a plan from a set of candidates with our criteria is beneficial, as it enforces a novelty-based exploration in the space of candidate logical plans. The role of prefixes and orderings in an LGP is captured correctly by our novelty measure, which outperforms classical plan similarity metrics like action set similarity (not shown in Tab. 1 due to space constraints), which is inaccurate in the context of LGP, where action ordering and precedence can not be neglected.

Analysis of Conflict Extraction Hypothesis: *"Metareasoning is faster than eager and lazy conflict extraction"*.

For *N=4*, we compare three different methods for extracting prefix conflicts: *eager* (finds minimal prefix using the *sequence* bound) *lazy-pose* (enhacement of *lazy* that checks only the *pose* bound to try to extract a conflict) and *meta* (metareasoning approach for conflict extraction).

Our metareasoning approach delivers a speedup across problems (*meta* is better in 12 vs *eager* 6). The *lazy-pose* (not shown in Tab. 1 due to space constraints) sometimes provides small infeasible prefixes with the *pose* bound, but is slower than *meta* and *eager*. Finally, note that sequence bounds of feasible NLPs are very fast to compute. This explains why in some problem *eager* is faster than *meta* even if it performs more geometric checks in total.

Conclusions

This paper is the first to propose a systematic interface between state-of-the-art symbolic planners and nonlinear constrained path optimization methods to solve Logic-Geometric Programs. A key idea of our approach is to efficiently identify geometric conflicts in the form of minimal infeasible action prefixes, and incorporate this information back into the symbolic planner through a multi-prefix forbidding compilation. Based on this general interface, we further developed a metareasoning strategy to minimize the number of calls to the motion planner, and a new novelty criteria for selecting plans from a set of candidates. Our approach systematically outperforms the baseline LGP solver, solving more problems and faster, especially when the solution requires long action sequences.

As future work, symbolic and geometric heuristics could be combined to guide the logical planner. Not only forbidding a tree of prefixes, as presented here, but informing the planner that some partial action sequences have been found to be geometrically feasible in previous iterations.

Acknowledgements

This research was supported by the German-Israeli Foundation for Scientific Research (GIF) grant I-1491-407.6/2019, the German Research Foundation (DFG) under Germany's Excellence Strategy EXC 2002/1–390523135 "Science of Intelligence", and the International Max-Planck Research School for Intelligent Systems (IMPRS-IS).

References

Bäckström, C.; and Nebel, B. 1995. Complexity Results for SAS$^+$ Planning. *Computational Intelligence*, 11(4): 625–655.

Baier, J. A.; and McIlraith, S. A. 2006. Planning with Temporally Extended Goals Using Heuristic Search. In *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling, ICAPS 2006*, 342–345. AAAI.

Bellman, R. 1957. *Dynamic Programming*. Dover Publications. ISBN 9780486428093.

Bidot, J.; Karlsson, L.; Lagriffoul, F.; and Saffiotti, A. 2017. Geometric backtracking for combined task and motion planning in robotic systems. *Artif. Intell.*, 247: 229–265.

Dantam, N. T.; Kingston, Z. K.; Chaudhuri, S.; and Kavraki, L. E. 2016. Incremental Task and Motion Planning: A Constraint-Based Approach. In *Robotics: Science and systems*, volume 12, 00052. Ann Arbor, MI, USA.

Ferrer-Mestres, J.; Francès, G.; and Geffner, H. 2017. Combined Task and Motion Planning as Classical AI Planning. *CoRR*, abs/1706.06927.

Garrett, C. R.; Chitnis, R.; Holladay, R.; Kim, B.; Silver, T.; Kaelbling, L. P.; and Lozano-Pérez, T. 2021. Integrated Task and Motion Planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1): 265–293.

Garrett, C. R.; Lozano-Pérez, T.; and Kaelbling, L. P. 2020. PDDLStream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning. In *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020*, 440–448. AAAI Press.

Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artif. Intell.*, 173(5-6): 619–668.

Katz, M.; and Sohrabi, S. 2020. Reshaping Diverse Planning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, 9892–9899. AAAI Press.

Katz, M.; Sohrabi, S.; Udrea, O.; and Winterer, D. 2018. A Novel Iterative Approach to Top-k Planning. In *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS 2018, Delft, The Netherlands, June 24-29, 2018*, 132–140. AAAI Press.

Lagriffoul, F.; Dimitrov, D.; Bidot, J.; Saffiotti, A.; and Karlsson, L. 2014. Efficiently combining task and motion planning using geometric constraints. *The International Journal of Robotics Research*, 33(14): 1726–1747.

LaValle, S. M. 2006. *Planning Algorithms*. Cambridge University Press. ISBN 9780511546877.

Lelis, L. H. S. 2013. Active Stratified Sampling with Clustering-Based Type Systems for Predicting the Search Tree Size of Problems with Real-Valued Heuristics. In *Proceedings of the Sixth Annual Symposium on Combinatorial Search, SOCS 2013*. AAAI Press.

Lipovetzky, N. 2021. Width-Based Algorithms for Common Problems in Control, Planning and Reinforcement Learning. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, 4956–4960. ijcai.org.

Richter, S.; and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39: 127–177.

Russell, S. J.; and Wefald, E. 1991. Principles of Metareasoning. *Artif. Intell.*, 49(1-3): 361–395.

Silva, J. P. M.; and Sakallah, K. A. 1999. GRASP: A Search Algorithm for Propositional Satisfiability. *IEEE Trans. Computers*, 48(5): 506–521.

Srivastava, S.; Fang, E.; Riano, L.; Chitnis, R.; Russell, S. J.; and Abbeel, P. 2014. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, 639–646. IEEE.

Tate, A. 1977. Generating Project Networks. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence. Cambridge, MA, USA, August 22-25, 1977*, 888–893. William Kaufmann.

Toussaint, M. 2015. Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015*, 1930–1936. AAAI Press.

Toussaint, M.; Allen, K. R.; Smith, K. A.; and Tenenbaum, J. B. 2018. Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning. In *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*.

Toussaint, M.; Ha, J.; and Driess, D. 2020. Describing Physics For Physical Reasoning: Force-Based Sequential Manipulation Planning. *IEEE Robotics Autom. Lett.*, 5(4): 6209–6216.

Toussaint, M.; and Lopes, M. 2017. Multi-bound tree search for logic-geometric programming in cooperative manipulation domains. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, 4044–4051. IEEE.

Tuisov, A.; and Katz, M. 2021. The Fewer the Merrier: Pruning Preferred Operators with Novelty. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, 4190–4196. ijcai.org.

Williams, B. C.; and Ragno, R. J. 2007. Conflict-directed A* and its role in model-based embedded systems. *Discret. Appl. Math.*, 155(12): 1562–1595.