Learning Efficient Abstract Planning Models that Choose What to Predict

Anonymous Author(s)
Affiliation
Address
email

Abstract: An effective approach to solving long-horizon tasks in robotics domains with continuous state and action spaces is bilevel planning, wherein a high-level search over an abstraction of an environment is used to guide low-level decision-making. Recent work has shown how to enable such bilevel planning by learning abstract models in the form of symbolic operators and neural samplers. In this work, we show that existing symbolic operator learning approaches fall short in many robotics domains where a robot's actions tend to cause a large number of irrelevant changes in the abstract state. This is primarily because they attempt to learn operators that exactly predict all observed changes in the abstract state. To overcome this issue, we propose to learn operators that 'choose what to predict' by only modelling changes *necessary* for abstract planning to achieve specified goals. Experimentally, we show that our approach learns operators that lead to efficient planning across 10 different hybrid robotics domains, including 4 from the challenging BEHAVIOR-100 benchmark, while generalizing to novel initial states, goals, and objects.

Keywords: Learning for TAMP, Abstraction Learning, Long-horizon Problems

1 Introduction

Solving long-horizon robotics problems in domains with continuous state and action spaces is extremely challenging, even when the transition function is deterministic and known. One effective strategy is to learn abstractions that capture the essential structure of the domain and then leverage hierarchical planning approaches like task and motion planning (TAMP) to solve new tasks. A typical approach is to first learn state abstractions in the form of symbolic *predicates* (classifiers on the low-level state, such as `InGripper`), then learn *operator descriptions* and *samplers* in terms of these predicates [1, 2]. The operators describe a partial transition model in the abstract space, while the samplers enable the search for realizations of abstract actions in terms of primitive actions. In this paper, we focus on the problem of learning operator descriptions from very few demonstrations given a set of predicates, an accurate low-level transition model, and a set of parameterized controllers (such as `Pick(x, y, z)`) that serve as primitive actions. We hope to leverage search-then-sample bilevel planning [3, 2, 4] using these operators to aggressively generalize to a highly variable set of problem domain sizes, initial states, and goals in challenging robotics tasks.

A natural objective for the problem of finding good abstract domain models is prediction accuracy [3, 1], which would be appropriate if we were using the abstract model to make precise predictions, but in fact we are using it as *planning guidance* for an accurate low-level model. So instead, our objective is to find an abstract model that maximally improves the performance of the bilevel planning algorithm, given the available data. The difference between these objectives is stark in many robotics domains where robot actions can affect many relationships among the robot and objects in the world. To make highly accurate predictions about all of these state changes might require a very fine-grained abstract model with many complex operators. Such a model would require a lot

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

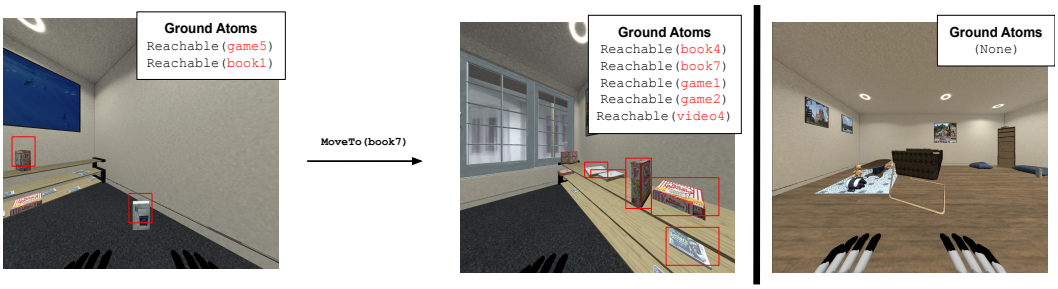

Figure 1: **Example demonstration transition and evaluation task from BEHAVIOR-100 task**. (Left) Visualization and high-level states for an example transition where the robot moves from being in range of picking a board game box and a book to picking 2 books, 2 board game boxes and a video game box. (Right) Visualization of an evaluation task where the robot starts out not in range of picking any objects.

of data to learn reliably, be very slow to plan with, and also be unlikely to generalize to novel tasks with different numbers and configurations of objects.

For example, consider the *Sorting Books* task from the BEHAVIOR-100 benchmark [5]. The goal is to retrieve a number of books strewn about a living room and place them on a shelf. A `Reachable(?object)` predicate is given to indicate when the robot is close enough to an object to pick it up. When the robot moves to pick a particular object, the set of objects that are reachable varies depending on their specific configuration. Figure 1 shows a transition where the robot moves to put itself in range of picking up `book7`, but happens to also be in range of picking up several other items. Optimizing prediction error on this transition would yield a very complex operator that is overfit to this specific situation and thus neither useful for efficient high-level planning nor generalizable to new tasks with different object configurations (e.g. the test situation depicted in the right panel of Figure 1, where the robot is initially not in reachable range of any objects).

Op–MoveToBook–Prediction–Error:
Args: ?objA ?objB ?objC ?objD ?objE ?objF ?objG
Preconditions: (**and** (Reachable ?objA) (Reachable ?objB))
Add Effects: (**and** (Reachable ?objC) (Reachable ?objD)(Reachable ?objE) (Reachable ?objF)(Reachable ?objG))
Delete Effects: (**and** (Reachable ?objA) (Reachable ?objB))

Op–MoveToBook–Necessary–Changes:
Args: ?objA
Preconditions: ()
Add Effects: (Reachable ?objA)
Delete Effects: ($\forall$?x. ?objA $\neq$?x $\Rightarrow$ (Reachable ?x))

In this work, we take seriously the objective of learning a set of operators that is tied to overall planning performance. We observe that, in order to generate useful high-level plans, *operators need only model predicate changes that are necessary for high-level search*. Optimizing this objective and carefully choosing what to predict allows us to learn operators that generalize better and yield faster planning.

Our main contributions are (1) the formulation of a novel objective for operator learning based on planning performance, (2) a procedure for distinguishing necessary changes within the high-level states of provided demonstrations, and (3) an algorithm that leverages (1) and (2) to learn symbolic operators from demonstrations via a hill-climbing search. We test our method on a wide range of complex robotic planning problems and find that our learned operators enable bilevel planning to solve challenging tasks and generalize substantially from a small number of examples.

2 Problem Setting

We aim to develop an efficient method for solving TAMP problems in complex domains with long-horizon solutions given structured low-level continuous state and action spaces [4, 1, 2]. We assume an 'object-oriented' state space: a state $x \in \mathcal{X}$ is characterized by the continuous proper-

ties (e.g. pose, color, material) of a set of objects. Actions, $u \in \mathcal{U}$, are short-horizon policies with both discrete and continuous parameters, which accomplish a desired change in state (e.g., `Pick(block, θ)` where θ is a grasp transform). Transitions are deterministic and a simulator $f : \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{X}$ predicts the next state given current state and action. The state and action representations, as well as the transition function can be acquired by engineering or learning [6, 4, 7].

PREIMAGEBACKCHAINING($(\Omega, (\overline{x}, \overline{u}), \mathcal{O}, g)$)

$n \leftarrow \text{length}(\overline{x})$; $\alpha_n \leftarrow g$
for $i \leftarrow n-1, n-2, \ldots, 0$ **do**
  $s_i, s_{i+1} \leftarrow$ ABSTRACT(x_i), ABSTRACT(x_{i+1})
  $\underline{\omega}_{\text{best}} \leftarrow$ FINDBESTCONSISTENTOP$(\Omega, s_i, s_{i+1}, \alpha_{i+1}, \mathcal{O})$
  if $\underline{\omega}_{best} = \textit{Null}$ **then**
    break
  $\underline{\omega_{i+1}} \leftarrow \underline{\omega}_{\text{best}}$
  $\alpha_i \leftarrow \underline{P_{i+1}} \cup (\alpha_{i+1} \setminus \underline{E^+_{i+1}})$
return $(\underline{\omega_{i+1}}, \ldots, \underline{\omega_n}), (\alpha_i, \ldots, \alpha_n)$

Algorithm 1: Preimage backchaining procedure (details in (§C)).

HILLCLIMBINGSEARCH($\mathcal{D}$)

$J_{\text{last}} \leftarrow \infty$; $J_{\text{curr}} \leftarrow \text{J}(\Omega, \mathcal{D})$
$\Omega \leftarrow \emptyset$
while $J_{curr} < J_{last}$ **do**
  $\Omega' \leftarrow$ IMPROVECOVERAGE$(\Omega, \mathcal{D})$
  if $\text{J}(\Omega', \mathcal{D}) < J_{curr}$ **then**
    $\Omega \leftarrow \Omega'$; $J_{\text{curr}} \leftarrow \text{J}(\Omega, \mathcal{D})$
  $\Omega' \leftarrow$ REDUCECOMPLEXITY$(\Omega, \mathcal{D})$
  if $\text{J}(\Omega', \mathcal{D}) < J_{curr}$ **then**
    $\Omega \leftarrow \Omega'$; $J_{\text{curr}} \leftarrow \text{J}(\Omega, \mathcal{D})$
  $J_{\text{last}} \leftarrow J_{\text{curr}}$; $J_{\text{curr}} \leftarrow \text{J}(\Omega, \mathcal{D})$
return Ω

Algorithm 2: HILLCLIMBINGSEARCH learns operators that optimize objective J.

Since planning in this low-level space can be expensive and unreliable [8, 1, 2], we pursue a search-then-sample bilevel planning strategy in which search in an abstraction is used to guide low-level planning. We assume we are given a set of *predicates* Ψ to define discrete properties of and relations between objects (e.g., `On`) via a classifier function that outputs true or false for a tuple of objects in a low-level state. Predicates induce a state abstraction ABSTRACT : $\mathcal{X} \rightarrow \mathcal{S}$ where ABSTRACT(x) is the set of true *ground atoms* in x (e.g., {`HandEmpty(robot), On(b1, b2), ...`}). The low-level state space, action space, and simulator together with the predicates comprise an *environment*. To enable bilevel planning for an environment, we must *learn* a partial abstract transition model over the predicates in the form of symbolic operators.

We consider a standard learning-from-demonstration setting where we are given a set of training tasks with demonstrations, and must generalize to some held-out test tasks. A *task* $T \in \mathcal{T}$ is characterized by a set of objects $\mathcal{O}$, an initial state $x_0 \in \mathcal{X}$, and a goal g. The goal is a set of ground atoms and is *achieved* in x if $g \subseteq$ ABSTRACT(x). A *solution* to a task is a sequence of actions $\overline{u} = (u_1, \ldots, u_n)$ that achieve the goal ($g \subseteq$ ABSTRACT(x_n), and $x_i = f(x_{i-1}, u_i)$ for $1 \leq i \leq n$) from the initial state. Each environment is associated with a *task distribution*. Our objective is to maximize the likelihood of solving tasks from this distribution within a planning time budget.

3 Operators for Bilevel Planning

Symbolic operators, defined in PDDL [9] to support efficient planning, specify a transition function over our state abstraction. Formally, an operator ω has *arguments* $\overline{v}$, *preconditions* P, *add effects* E^+, *delete effects* E^-, and a *controller* C. The preconditions and effects are each expressions over the arguments that describe conditions under which the operator can be executed (e.g. `Reachable(?book1)`) and resulting changes to the abstract state (e.g. `Holding(?book1)`) respectively. The controller is a policy (from the environment's action space), parameterized by some of the discrete operator arguments, as well as continuous parameters whose values will be chosen during the sampling phase of bilevel planning. A substitution of arguments to objects induces a *ground operator* $\underline{\omega} = \langle \underline{P}, \underline{E^+}, \underline{E^-}, \underline{C} \rangle$. Given a ground operator $\underline{\omega} = \langle \underline{P}, \underline{E^+}, \underline{E^-}, \underline{C} \rangle$, if $\underline{P} \subseteq s$, then the successor abstract state s' is $(s \setminus \underline{E^-}) \cup \underline{E^+}$ where $\setminus$ is set difference. We use $F(s, \underline{\omega}) = s'$ to denote this (partial) abstract transition function.

In previous work on operator learning for bilevel planning [3, 2], operators are connected to the underlying environment via the following semantics: if $F(s, \underline{\omega}) = s'$, then *there exists* some low-level transition (x, u, x') where ABSTRACT$(x) = s$, ABSTRACT$(x') = s'$, and $u = \underline{C}(\theta)$ for some θ.

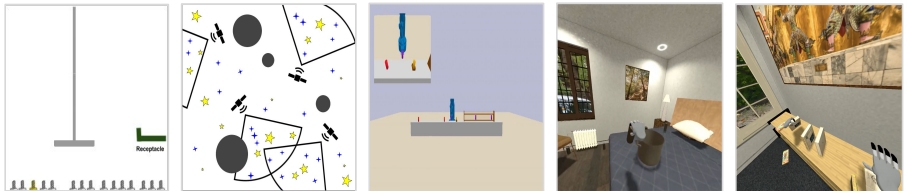

Figure 2: **Environments**. Visualizations for *Screws*, *Satellites*, *Painting*, *Collecting Cans*, and *Sorting Books*.

These semantics embody the "prediction error" view in that ABSTRACT must predict the entire next state ($\text{ABSTRACT}(x') = s'$). Towards implementing the alternative "necessary changes" view, we will instead only require the abstract state output by F to be a *subset* of the atoms in the next state, that is, $\text{ABSTRACT}(x') \subseteq s'$. Thus, we permit our operators to have universally quantified single-predicate delete effects (e.g., `∀?v. Reachable(?v)`). Intuitively, F is now responsible only for predicting abstract *subgoals* to guide low-level planning, rather than predicting entire successor abstract states. Under these new semantics, F encodes a *nondeterministic* abstract transition model, where the output atoms s' represent the set of possible abstract states where these atoms hold.

Given operators, we can solve new tasks via search-then-sample bilevel planning (decribed in detail in appendix (§A) and previous work [10, 2, 3, 4, 1]). Given the task goal g, initial state x_0, and corresponding abstract state $s_0 = \text{ABSTRACT}(x_0)$, bilevel planning uses AI planning techniques (e.g., Helmert [11]) to generate candidate *abstract plans*. An abstract plan is a sequence of ground operators $(\underline{\omega}_1, \ldots, \underline{\omega}_n)$ where $g \subseteq s_n$ and $s_i = F(s_{i-1}, \underline{\omega}_i)$ for $1 \leq i \leq n$. The corresponding abstract state sequence $(s_1, \ldots, s_n)$ serves as a sequence of *subgoals* for low-level planning. The controller sequence $(\underline{C}_1, \ldots, \underline{C}_n)$ provides a *plan sketch*, where all that remains is to *refine* the plan by "filling in" the continuous parameters. We sample continuous parameters θ for each controller starting from the first and checking if the controller achieves its corresponding subgoal s_i. If we cannot sample such parameters within a constant time budget, we backtrack and resample, and eventually even generate a new abstract plan.

4 Learning Operators from Demonstrations

To enable efficient bilevel planning, operators must generate abstract plans corresponding to plan sketches that are likely to be refinable into low-level controller executions. Given demonstrations $\mathcal{D}$ consisting of a goal g, action sequence $\overline{u}$, corresponding state sequence $\overline{x}$, and set of objects $\mathcal{O}$, we wish to find the simplest set of operators such that for every training task, abstract planning with these operators is able to generate the plan sketch corresponding to the demonstration in $\mathcal{D}$.

Specifically, we minimize the following objective:

$$J(\Omega) \triangleq (1 - \texttt{coverage}(\mathcal{D}, \Omega)) + \lambda \texttt{complexity}(\Omega) \tag{1}$$

where `coverage` is the fraction of demonstration plan sketches that planning with operator set Ω is able to produce and `complexity` is the number of operators. In our experiments, we set λ to be small enough so that complexity is never decreased at the expense of coverage.

We will approach this optimization problem using a hill-climbing search (Algorithm 2), which benefits from having a more fine-grained interpretation of coverage defined over *transitions* within a trajectory instead of entire trajectories. To develop this definition, we will first introduce the notion of *necessary atoms*, which are the conditions that must hold true for an abstract plan suffix to legally achieve the goal.

Definition 4.1. Given an operator set Ω, and an abstract plan $(\underline{\omega}_1, \ldots, \underline{\omega}_n)$ in terms of Ω that achieves goal g, the *necessary atoms* at step n are $\alpha_n \triangleq g$, and at step $0 \leq i < n$ are $\alpha_i \triangleq \underline{P_{i+1}} \cup (\alpha_{i+1} \setminus \underline{E^+_{i+1}})$.

If operators in Ω model each of the necessary atoms for every timestep, then these operators must be sufficient for producing the corresponding abstract plan via symbolic planning. Moreover, each necessary atom set is minimal in that no atoms can be removed without violating either the goal

or a future operator's preconditions. Thus, we need only learn operators to model changes in these necessary atoms.

Given necessary atoms, we can now define what it means for some sequence of operators to be *consistent* with some part of a demonstration.

Definition 4.2. An abstract plan suffix $(\underline{\omega}_k, \ldots, \underline{\omega}_n)$ using operators from set Ω and with necessary atoms $(\alpha_{k-1}, \ldots, \alpha_n)$ is *consistent* with a demonstration $(\overline{x}, \overline{u})$ for goal g and timesteps $1 \leq k \leq n$, where $\overline{x} = (x_0, \ldots, x_n)$ and $\overline{u} = (u_1, \ldots, u_n)$ if for $k \leq i \leq n$, (1) the states are consistent: $\alpha_i \subseteq F(\text{ABSTRACT}(x_{i-1}), \underline{\omega}_i) \subseteq \text{ABSTRACT}(x_i)$; and (2) the actions are consistent: if the controller for $\underline{\omega}_i$ is $\underline{C}_i$, then $u_i = \underline{C}_i(\theta)$ for some θ.

Note that consistency is defined at the level of individual transitions within a demonstration. Thus, it is useful even if Ω does not contain sufficient operators to produce a plan that is consistent with an entire demonstration. Given this notion, we can now define *coverage* for a particular demonstration.

Definition 4.3. The *demonstration coverage*, $\eta(\Omega, (\overline{x}, \overline{u}, g, \mathcal{O}))$ of demonstration $(\overline{x}, \overline{u})$ of length n for goal g and objects $\mathcal{O}$ by operators Ω, is a number from 0 to n indicating the length of the longest consistent suffix for the demonstration that can be generated with Ω ground with objects $\mathcal{O}$.

Now, we define a fine-grained notion of coverage for optimization via hill-climbing:

$$\texttt{coverage}(\Omega, \mathcal{D}) = \Sigma_{(\overline{x}, \overline{u}, g, \mathcal{O}) \in \mathcal{D}} \frac{\eta(\omega, (\overline{x}, \overline{u}, g, \mathcal{O}))}{|\overline{u}|} \tag{2}$$

Computing `coverage` implicitly requires that we compute an abstract plan suffix, as well as necessary atoms (in order to compute η). To do this efficiently, we will leverage the fact that we know the necessary atoms for the final timestep of every demonstration (they are simply the goal atoms!), and compute necessary atoms and plan suffixes for previous timesteps via a *preimage backchaining* procedure that starts at the goal and works backwards given operators in Ω. Pseudocode for this algorithm is shown in Algorithm 1. We pass backward through each transition in a demonstration trajectory while attempting to choose an operator in Ω to cover (per Definition 4.2) the transition and then updating the transition's necessary atoms using Definition 4.1. If there are multiple operators in Ω that all satisfy the conditions of Definition 4.2, we use a heuristic that selects the operator that best matches this particular transition (see (§C) in the appendix for details).

Hill-Climbing Search. We perform a hill-climbing search on $J(\Omega)$ (Equation 1), starting with the empty operator set. We have two search-step generators: the IMPROVECOVERAGE generator picks a demonstration that is not completely covered by the current operator set and proposes a change that will increase the `coverage` term of the objective, and the REDUCECOMPLEXITY generator simply proposes to delete operators to decrease the `complexity` term.

IMPROVECOVERAGE generator. Given a candidate operator set Ω, preimage backchaining is used to identify an abstract plan suffix of length k with corresponding necessary atoms $(\alpha_{k-1}, \ldots, \alpha_n)$ that is consistent with a demonstration $(\overline{x}, \overline{u})$ for goal g, where $\overline{x} = (x_0, \ldots, x_n)$ and $\overline{u} = (u_1, \ldots, u_n)$, and where $1 \leq k < n$. Since $k < n$, we know the transition $(x_{k-2}, u_{k-1}, x_{k-1})$ with necessary atoms α_{k-1} is not covered by any operator in Ω. To improve the `coverage` term in our objective, we wish to generate at least one new operator ω, and potentially modify existing operators to cover this transition without uncovering any others. Definition 4.2 gives us the following constraints on the new operator ω with grounding $\underline{\omega}$:

1. The controller $\underline{C}$ must match u_{k-1}.
2. $\underline{P} \subseteq s_{k-2}$, i.e., the preconditions must be satisfied in the previous state.
3. $((s_{k-2} \setminus s_{k-1}) \cap \alpha_{k-1}) \subseteq \underline{E}^+$, i.e., the add effects must include the added necessary atoms.
4. $F(s_{k-2}, \underline{\omega}) \subseteq s_{k-1}$, i.e., the delete effects should remove all atoms in s_{k-2} but not in s_{k-1}.
5. $\overline{v}$ must at least include one variable (with corresponding type) for each of the objects in 1–4.

These constraints exactly determine ω's controller, but there are many possible choices for the other components that would satisfy 2–5. One straightforward choice would be to optimize prediction error over the single transition $(s_{k-2}, u_{k-1}, s_{k-1})$ and necessary atoms α_{k-1}, but this would lead to a hyper-specific operator.

Environment	Ours	LOFT	LOFT+Replay	CI	CI + QE	GNN Shoot	GNN MF
Painting	**98.80 (0.42)**	0.00 (0.00)	**98.20 (0.91)**	**99.00 (0.31)**	93.40 (1.47)	36.00 (3.39)	0.60 (0.29)
Satellites	**93.40 (3.52)**	0.00 (0.00)	34 (5.28)	**91.60 (2.68)**	**95.20 (1.30)**	40.40 (3.04)	11.00 (1.44)
Cluttered 1D	**100.00 (0.00)**	17.20 (5.46)	0.00 (0.00)	17.40 (5.52)	92.80 (0.90)	98.60 (0.63)	98.60 (0.63)
Screws	**100.00 (0.00)**	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	50.00 (15.81)	95.60 (3.05)	95.80 (3.07)
Cluttered Satellites	**95.20 (0.75)**	0.00 (0.00)	0.00 (0.00)	1.60 (0.61)	6.00 (1.57)	4.80 (1.27)	0.00 (0.00)
Cluttered Painting	**99.20 (0.42)**	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	4.60 (1.16)	0.00 (0.00)
Opening Presents	**100.00 (0.00)**	0.00 (0.00)	-	83.00 (10.77)	83.00 (10.77)	28.00 (5.96)	0.00 (0.00)
Locking Windows	**100.00 (0.00)**	0.00 (0.00)	-	90.00 (4.47)	88.00 (4.42)	0.00 (0.00)	0.00 (0.00)
Collecting Cans	**77.00 (11.75)**	0.00 (0.00)	-	0.00 (0.00)	1.00 (0.94)	0.00 (0.00)	0.00 (0.00)
Sorting Books	**69.00 (11.61)**	0.00 (0.00)	-	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)

Table 1: Percentage success rate on test tasks for all domains. Note that BEHAVIOR domains use training and testing set sizes of 10 tasks, while all other domains use training and testing set sizes of 50 tasks. The percentage standard error is shown in brackets. All entries within one standard error from the best mean success rate are shown in bold.

Instead, we construct a more general operator that covers this transition as well as other transitions in the data set. Specifically, we set ω's ground controller $\underline{C} = u_{k-1}$ to satisfy condition 1 above, and ground add effects $\underline{E^+} = ((s_{k-2} \setminus s_{k-1}) \cap \alpha_{k-1})$ to satisfy condition 3. We then construct the controller C and add effects E^+ by 'lifting': i.e, replacing all objects that appear in $\underline{C}$ and $\underline{E^+}$ by variables of the same type. We record the substitution necessary between the variables and objects for this transition as δ_τ. The operator's arguments are then the union of the variables that appear here. To find preconditions and delete effects that satisfy conditions 2 and 4, we will use each substitution δ_τ associated with each transition in $\mathcal{D}_\omega$ to *lift* each transition τ, replacing all objects with arguments while discarding any atom with objects not in δ_τ. Following Chitnis et al. [2], the *preconditions* are then: $P \leftarrow \bigcap_{\tau=(s_i,\cdot,\cdot)\in\mathcal{D}_\omega} \delta_\tau(s_i)$. Similarly, the *atomic delete effects* are: $E_\circ^- \leftarrow \bigcup_{\tau=(s_i,\cdot,s_{i+1})\in\mathcal{D}_\omega} \delta_\tau(s_{i+1}) \setminus \delta_\tau(s_i)$. Finally, the operator's *quantified delete effects* are set to be the set of predicates that appear in the current predicted abstract state, but not the observed abstract state: $F(s_i, \underline{\omega}) \setminus s_{i-1}$. We obtain the final delete effects by simply combining the atomic and quantified effects.

Some additional bookkeeping is necessary once this new operator has been induced to guarantee that this generator decreases the coverage term (we must recompute preconditions and delete effects for *all* current candidate operators, since their respective transition datasets might have changed, etc.), and we detail these, along with a proof of termination, in the appendix (§B.1).

REDUCECOMPLEXITY generator. Given a candidate operator set Ω, we simply delete a single operator from the set and then recompute preconditions and delete effects for all remaining operators as described above (since now, their associated transition datasets $\mathcal{D}_\omega$ might change).

5 Experimental Results

Our experiments are designed to empirically answer the following questions:

- **Q1.** Does our approach learn operators capable of generalizing to novel goals and situations with different objects than seen during training?
- **Q2.** How effective is bilevel planning using our learned operators vs. existing baselines?
- **Q3.** How does the performance of our approach vary with the amount of training data?

For additional analyses, including the complexity of learned operators, learning and task planning efficiency, and ablations of our method, see (§G) and (§H) in the appendix.

Environments. We provide high-level environment details, with specifics in the appendix (§E). We deliberately include a few simple environments to highlight differences between methods.

- *Painting*: a challenging robotics environment used by Silver et al. [3, 1]. A robot in 3D must pick, wash, dry, paint, and then place various objects.
- *Satellites*: a 2D environment inspired by the Satellites domain from Bacchus [12], but augmented with collisions and realistic sensing constraints. See appedix (§E) for further details.
- *Cluttered 1D*: a simple environment where the robot must move and collect objects cluttered along a 1D line. An object can only be collected if it is reachable.

- *Screws*: a 2D environment where the agent controls a magnetic crane and must pick specific screws from clutter to place them into a receptacle.
- *Cluttered Satellites*: same as "Satellites", except readings must be taken from multiple objects.
- *Cluttered Painting*: same as Painting, except the robot can be next to many objects at a time.
- *BEHAVIOR-100 Tasks*: a set of complex, long-horizon household robotic tasks simulated with realistic 3D models of objects and homes [5]. In *Opening Presents*, the robot must open a number of boxes. In *Locking Windows*, the robot must close a number of open windows. In *Collecting Cans*, the robot must pick up a number of empty soda cans strewn amongst the house and throw them into a trash can. In *Sorting Books*, the robot must find books in a living room and place them each onto a cluttered shelf.

Approaches. We describe the approaches we compare against, with details in the appendix (§F).

- *Cluster and Intersect (CI)* [1]: Induces a different operator for every set of unique lifted effects.
- *LOFT* [3]: Optimizes prediction error, but with a more general class of preconditions. We include a version that only learns from demonstrations (LOFT) and a version that collects additional transitions (2500) in each domain (LOFT+Replay) as in the original work. Collecting replay data in BEHAVIOR was intractable due to the size and complexity of the simulation.
- *Cluster and Intersect with Quantified Delete Effects (CI + QE)*: A variant of Cluster and Intersect that is capable of learning operators that have both atomic and quantified delete effects. It first runs Cluster and Intersect, and then induces quantified delete effects by optimizing prediction error via a hill-climbing search.
- *GNN Shooting*: Trains an abstract GNN policy with behavioral cloning and uses it for trajectory optimization. This is inspired by a baseline from [1], see (§F) in the appendix for details.
- *GNN Model-Free*: Uses the same trained GNN as above, but directly executes the policy.

Experimental Setup. For the non-BEHAVIOR environments, we run all methods on up to 50 training demonstrations. For BEHAVIOR environments, we use 10, since collecting training data in these complex environments is very time and memory intensive. Training demonstrations were collected by a hand-coded 'oracle' policy that was specific to each environment. All experiments were conducted on a quad-core Intel Xeon Platinum 8260 processor with a 192GB RAM limit, and all results are averaged over 10 random seeds. For each seed, we sample a set of *evaluation tasks* from the task distribution $\mathcal{T}$. *The evaluation tasks have more objects, different initial states, and more atoms in the goal than seen during training.* Our key measure of effective bilevel planning is success rate within a timeout (10 seconds for non-BEHAVIOR environments, 500 seconds for 3 BEHAVIOR environments, and 1500 seconds for "Sorting Books": the most complex environment).

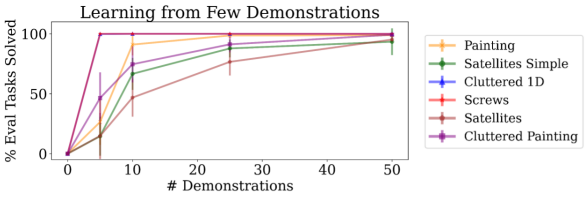

Figure 3: Data-efficiency of main approach.

Results and Analysis. Table 1 shows the success rate achieved on testing tasks for all our environments and methods. Our method solves many more held-out tasks within the timeout than the baselines, demonstrating its efficacy even in environments where an assumption on downward refinablity might not hold. In our simpler environments (Painting and Satellites; Opening Presents and Locking Windows for BEHAVIOR), the controllers cause a small number of changes in the abstract state, and baseline approaches (CI, CI + QE) perform reasonably well. In all the other environments, the controllers cause a large number of changes in the abstract state, and the performance of operator learning baselines degrades substantially, though GNN-baselines perform well on Cluttered 1D and Screws. Despite the increased complexity, our approach learns operators that are resilient to the lack of downward refinability, enabling bilevel planning to achieve a substantial test-time success

rate under timeout. The performance in Collecting Cans and Sorting Books is especially notable; all baselines achieve a negligible success rate, while our approach achieves a near 70% rate on testing tasks. Upon investigation, we found that failures are due to local minima during learning for certain random seeds. Figure 3 shows our method's testing success rate as a function of the size of its training set for our non-BEHAVIOR environments. Our approach's performance improves with more data, though as the dataset size increases, the impact of additional data on performance reduces.

6 Related Work

Our work continues a long line of research in learning operators for planning [13, 14, 15, 16, 17, 18, 19, 20, 21]; see Arora et al. [22] for a recent survey. This previous work focuses on learning operators from purely discrete plan traces in the context of classical (not bilevel) planning.

Other work has considered learning symbolic planning models in continuous environments [23, 24, 25, 26, 27, 28, 29, 30, 31], but typically the interface between symbolic planner and low-level policies assumes downward refinability, which requires that every valid high-level plan must be refinable into low-level steps [32], a critical assumption we do not make. Therefore, our efforts are most directly inspired by LOFT [3] and learning Neuro-Symbolic Relational Transition Models [2], which optimize prediction error to learn operators for bilevel planning. Like our method, LOFT performs a search over operator sets, but commits to modeling all effects seen in the data and searches only over operator preconditions. We point out the limitations of optimizing prediction error in complex robotics environments. We include LOFT and Cluster and Intersect as baselines representative of these previous methods in our experiments.

The bilevel planner used in our work can be viewed as a search-then-sample solver for (TAMP) [33, 34, 10]. This bilevel strategy allows for fast planning in continuous state and action spaces, while avoiding the downward refinability assumption. To that end, our work also contributes to a recent line of work on learning for TAMP. Other efforts in this line include sampler learning [35, 36, 37, 38], heuristic learning [39, 40, 41], and abstract plan feasibility estimation [42, 43].

7 Limitations, Conclusions and Future Work

Our method assumes that the provided predicates Ψ comprise a good state abstraction given the task distribution for operator learning. With completely random or meaningless predicates, our approach is likely to learn very complex operators such that planning with these is unlikely to outperform non-symbolic behavior-cloning baselines. Fortunately, prior work [1] suggests such 'good' predicates can themselves be learned from data. There is no guarantee that our overall hill-climbing procedure will converge quickly; the IMPROVECOVERAGE successor generator is especially computationally expensive to run. In practice, we find its learning time to be comparable or faster than baseline methods in our domains ((§G) in the appendix), though this may not hold in more complex domains where a very large (e.g. greater than 100) number of operators need to be learned. Additionally, our IMPROVECOVERAGE successor generator is rather complicated, and there is perhaps a simpler way to achieve the same desiderata (i.e, generating a new operator set that is guaranteed to decrease the `coverage` term of our objective).

Overall, we proposed a new objective for operator learning that is specifically tailored to bilevel planning, and a search-based method for optimizing this objective. Experiments confirmed that operators learned with our new method lead to substantially better generalization and planning than those learned by optimizing prediction error and other reasonable baselines. Important next steps include learning all components necessary to enable bilevel planning, including the predicates [1], controllers [4], and object-oriented state-space, as well as handling stochasticity and partial observability. We believe that pursuing these steps will yield important progress toward solving sparse-feedback, long-horizon decision-making problems via modular learning at scale.

References

[1] T. Silver, R. Chitnis, N. Kumar, W. McClinton, T. Lozano-Pérez, L. P. Kaelbling, and J. Tenenbaum. Predicate invention for bilevel planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[2] R. Chitnis, T. Silver, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling. Learning neuro-symbolic relational transition models for bilevel planning. In *The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[3] T. Silver, R. Chitnis, J. Tenenbaum, L. P. Kaelbling, and T. Lozano-Pérez. Learning symbolic operators for task and motion planning. In *The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[4] T. Silver, A. Athalye, J. B. Tenenbaum, T. Lozano-Perez, and L. P. Kaelbling. Learning neuro-symbolic skills for bilevel planning. In *Conference on Robot Learning (CoRL)*, 2022.

[5] S. Srivastava, C. Li, M. Lingelbach, R. Martín-Martín, F. Xia, K. E. Vainio, Z. Lian, C. Gokmen, S. Buch, K. Liu, S. Savarese, H. Gweon, J. Wu, and L. Fei-Fei. BEHAVIOR: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on Robot Learning (CoRL)*, 2021.

[6] J. Brady, R. S. Zimmermann, Y. Sharma, B. Schölkopf, J. von Kügelgen, and W. Brendel. Provably learning object-centric representations. *The International Conference on Machine Learning (ICML)*, 2023.

[7] M. Chang, T. Ullman, A. Torralba, and J. Tenenbaum. A compositional object-based approach to learning physical dynamics. In *The International Conference on Learning Representations (ICLR)*, 2017. URL https://openreview.net/forum?id=Bkab5dqxe.

[8] T. Kurutach, A. Tamar, G. Yang, S. J. Russell, and P. Abbeel. Learning plannable representations with causal infogan. *Advances in Neural Information Processing Systems*, 31, 2018.

[9] A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. SRI, A. Barrett, D. Christianson, et al. PDDL— the planning domain definition language. *Technical Report, Tech. Rep.*, 1998.

[10] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 2021.

[11] M. Helmert. The fast downward planning system. *The Journal of Artificial Intelligence Research (JAIR)*, 2006.

[12] F. Bacchus. Aips 2000 planning competition: The fifth international conference on artificial intelligence planning and scheduling systems. *AI magazine*, 2001.

[13] G. L. Drescher. *Made-up minds: a constructivist approach to artificial intelligence*. MIT press, 1991.

[14] E. Amir and A. Chang. Learning partially observable deterministic action models. *The Journal of Artificial Intelligence Research (JAIR)*, 2008.

[15] N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrčen, et al. Object–action complexes: Grounded abstractions of sensory–motor processes. *Robotics and Autonomous Systems*, 2011.

[16] T. Lang, M. Toussaint, and K. Kersting. Exploration in relational domains for model-based reinforcement learning. *The Journal of Machine Learning Research (JMLR)*, 2012.

[17] K. Mourao, L. S. Zettlemoyer, R. Petrick, and M. Steedman. Learning strips operators from noisy and incomplete observations. *arXiv preprint arXiv:1210.4889*, 2012.

[18] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling. Learning symbolic models of stochastic domains. *The Journal of Artificial Intelligence Research (JAIR)*, 2007.

[19] C. Rodrigues, P. Gérard, C. Rouveirol, and H. Soldano. Active learning of relational action models. In *International Conference on Inductive Logic Programming*, 2011.

[20] S. N. Cresswell, T. L. McCluskey, and M. M. West. Acquiring planning domain models using LOCM. *The Knowledge Engineering Review*, 2013.

[21] D. Aineto, S. Jiménez, and E. Onaindia. Learning strips action models with classical planning. In *The International Conference on Automated Planning and Scheduling (ICAPS)*, 2018.

[22] A. Arora, H. Fiorino, D. Pellier, M. Métivier, and S. Pesty. A review of learning planning action models. *The Knowledge Engineering Review*, 2018.

[23] N. Jetchev, T. Lang, and M. Toussaint. Learning grounded relational symbols from continuous data for abstract reasoning. In *ICRA Workshop on Autonomous Learning*, 2013.

[24] E. Ugur and J. Piater. Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning. In *The IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[25] A. Ahmetoglu, M. Y. Seker, J. Piater, E. Oztop, and E. Ugur. Deepsym: Deep symbol generation and rule learning from unsupervised continuous robot interaction for planning. *arXiv preprint arXiv:2012.02532*, 2020.

[26] M. Asai and A. S. Fukunaga. Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary. In *The AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[27] B. Bonet and H. Geffner. Learning first-order symbolic representations for planning from the structure of the state space. In *The European Conference on Artificial Intelligence (ECAI)*, 2020.

[28] M. Asai and C. Muise. Learning neural-symbolic descriptive planning models via cube-space priors: The voyage home (to STRIPS). *arXiv preprint arXiv:2004.12850*, 2020.

[29] E. Umili, E. Antonioni, F. Riccio, R. Capobianco, D. Nardi, and G. De Giacomo. Learning a symbolic planning domain through the interaction with continuous environments. *ICAPS PRL Workshop*, 2021.

[30] G. Konidaris, L. P. Kaelbling, and T. Lozano-Pérez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *The Journal of Artificial Intelligence Research (JAIR)*, 2018.

[31] A. Curtis, T. Silver, J. B. Tenenbaum, T. Lozano-Pérez, and L. Kaelbling. Discovering state and action abstractions for generalized task and motion planning. In *The AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

[32] B. Marthi, S. J. Russell, and J. A. Wolfe. Angelic semantics for high-level actions. In *The International Conference on Automated Planning and Scheduling (ICAPS)*, pages 232–239, 2007.

[33] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *The IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[34] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki. Incremental task and motion planning: A constraint-based approach. In *Robotics: Science and Systems (R:SS)*, 2016.

[35] R. Chitnis, D. Hadfield-Menell, A. Gupta, S. Srivastava, E. Groshev, C. Lin, and P. Abbeel. Guided search for task and motion plans using learned heuristics. In *The IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[36] A. Mandalika, S. Choudhury, O. Salzman, and S. Srinivasa. Generalized lazy search for robot motion planning: Interleaving search and edge evaluation via event-based toggles. In *The International Conference on Automated Planning and Scheduling (ICAPS)*, 2019.

[37] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez. Learning compositional models of robot skills for task and motion planning. *The International Journal of Robotics Research (IJRR)*, 2021.

[38] J. Ortiz-Haro, J.-S. Ha, D. Driess, and M. Toussaint. Structured deep generative models for sampling on constraint manifolds in sequential manipulation. In *Conference on Robot Learning (CoRL)*, 2022.

[39] W. Shen, F. Trevizan, and S. Thiébaux. Learning domain-independent planning heuristics with hypergraph networks. In *The International Conference on Automated Planning and Scheduling (ICAPS)*, 2020.

[40] B. Kim and L. Shimanuki. Learning value functions with relational state representations for guiding task-and-motion planning. *Conference on Robot Learning (CoRL)*, 2019.

[41] C. Paxton, V. Raman, G. D. Hager, and M. Kobilarov. Combining neural networks and tree search for task and motion planning in challenging environments. In *The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[42] D. Driess, J.-S. Ha, and M. Toussaint. Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image. In *Robotics: Science and Systems (R:SS)*, 2020.

[43] M. Noseworthy, C. Moses, I. Brand, S. Castro, L. Kaelbling, T. Lozano-Pérez, and N. Roy. Active learning of abstract plan feasibility. In *Robotics: Science and Systems (R:SS)*, 2021.

[44] M. Helmert and C. Domshlak. Landmarks, critical paths and abstractions: what's the difference anyway? In *The International Conference on Automated Planning and Scheduling (ICAPS)*, 2009.

[45] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.