Action Space Reduction for Planning Domains

Harsha Kokel,[1]* Junkyu Lee,[2] Michael Katz,[2] Shirin Sohrabi,[2] Kavitha Srinivas[2]

[1] The University of Texas at Dallas
[2] IBM Research
hkokel@utdallas.edu, {Junkyu.Lee,michael.katz1,kavitha.srinivas}@ibm.com, ssohrab@us.ibm.com

Abstract

When solving planning problems with Reinforcement Learning (RL) algorithms, great care should be taken in defining action spaces. A naive translation of the planning action space incurs severe degradation in sample complexity, and thus in practice action spaces are often engineered manually in a domain-specific manner. Here, we propose an automated way of reducing the action spaces, by leveraging lifted mutex groups. Our experiments show a significant reduction in the action space size of the RL environments, across all tested planning domains, improving sample complexity of RL agents.

Introduction

AI planning tasks are commonly described in the planning domain definition language (PDDL) (McDermott 2000) (RL) tasks are often formulated as a Markov Decision Process (MDP). When planning problems are tackled with RL approaches, these problems can and often are formulated as an MDP. The state space of this MDP is defined over the set of ground fluents and the action space is defined over the set of ground operators. As the number of objects in the planning problem increases, the MDP state and action spaces grow significantly. While RL approaches are resilient to large state spaces, large discrete action spaces are problematic (Zahavy et al. 2018; Pazis and Parr 2011).

To remedy the large discrete action spaces of planning tasks, researchers either design the action space of the MDP manually from scratch (e.g., Dzeroski, Raedt, and Driessens (2001)) or modify the grounding process of the PDDL operators to generate reduced actions. In this work we focus on the latter approach. Intuitively, the number of ground operators defined by a PDDL operator is dictated by the number of parameters in that operator. If some parameters of an operator are removed, the number of ground operators generated by that operator would reduce. Building up on this intuition Silver and Chitnis (2020) propose to reduce the number of actions by eliminating the *manually annotated inessential parameters* while grounding.

Consider a gripper domain (McDermott 2000) where a robot moves balls between two rooms. Figure 1 depicts a sample initial state and the PDDL task in this domain. Here, for the schematic operator `drop(?b:ball, ?r:room, ?g:gripper)`, the parameters `?b` and `?r` are *inessential*—they are not required by the RL agent—as the ball `?b` and the room `?r` can be inferred from the selected gripper. That is, the ball carried by the gripper is `?b` and, the current location of the robot is `?r`. So, only the parameter `?g` is essential and rest can be dropped while defining the MDP action. PDDL operators encode transition dynamics and hence require all the parameters occurring in the preconditions and effects. Whereas, MDP actions are mere labels and do not encode transition dynamics. Hence, parameters required by the PDDL operator are not necessary for MDP actions. To make this distinction explicit, we refer to the action space of MDP as label set.

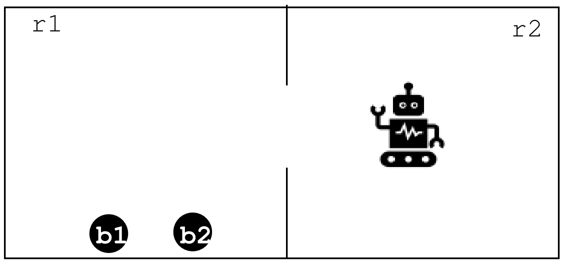
In this paper, we prove that existing methods for discovering lifted mutex groups can be leveraged to automatically identify the inessential parameters of the operators effectively. Our contributions are as follows. We propose an automated way of reducing the labels by defining the notion of valid label reduction and applicable operator mutex groups. We formally define the problem of obtaining a parameter seed set. Next, we propose to solve this problem by translating it to delete-free planning task and prove that the solution obtained is a valid label reduction. Then we empirically evaluate our approach on 14 STRIPS domains and show that it achieves significant reduction in action labels. And finally, we empirically show that the label reduction significantly improves the sample efficiency of standard RL agents.

Preliminaries

In this section, we first introduce the notations for a normalized PDDL task, then describe the semantics behind converting a PDDL task to MDP and provide brief overview of the lifted mutex groups that we use in our proposed approach.

PDDL task We follow the notation of Röger, Sievers, and Katz (2018) for normalized PDDL tasks, dropping axioms and conditional effects for simplicity. A *normalized PDDL task* $\Pi = \langle \mathcal{L}, \mathcal{O}, I, G \rangle$ is defined over a first-order language $\mathcal{L}$ that consists of a finite number of fluent predicates, variables, and objects. For formulas over $\mathcal{L}$ the *free variables* are defined as usual in first-order logic. Formulas not containing any free variables are called *grounded*. Otherwise, they are

*This work was partially done when HK was at IBM Research.



(a)

A normalized PDDL Gripper task $\Pi = \langle \mathcal{L}, \mathcal{O}, I, G \rangle$.

- Language $\mathcal{L}$ includes:
  typed objects: $\texttt{r1}, \texttt{r2}, \texttt{b1}, \texttt{b2}, \texttt{g1}, \texttt{g2}$ and
  predicates: $\texttt{at\_robby}(?\texttt{r}), \texttt{at}(?\texttt{b}, ?\texttt{r}),$
  $\texttt{free}(?\texttt{g}), \texttt{carry}(?\texttt{b}, ?\texttt{g})$
- Initial state I is: $\{\texttt{at}(\texttt{b1}, \texttt{r1}), \texttt{at}(\texttt{b2}, \texttt{r1}),$
  $\texttt{at\_robby}(\texttt{r2}), \texttt{free}(\texttt{g1}), \texttt{free}(\texttt{g2})\}$
- Goal state G is: $\{\texttt{at}(\texttt{b2}, \texttt{r2})\}$
- Schematic operators $\mathcal{O}$ consists:

```
move
  : params {?from : room, ?to : room}
  : pre {at_robby(?from)}
  : add {at_robby(?to)}
  : del {at_robby(?from)}

pick
  : params {?b : ball, ?r : room, ?g : gripper}
  : pre {at(?b, ?r), at_robby(?r),
      free(?g)}
  : add {carry(?b, ?g)}
  : del {at(?b, ?r), free(?g)}

drop
  : params {?b : ball, ?r : room, ?g : gripper}
  : pre {carry(?b, ?g), at_robby(?r)}
  : add {at(?b, ?r), free(?g))}
  : del {carry(?b, ?g)}
```

(b)

Figure 1: A running example of Gripper task, (a) an initial state with balls b1 and b2 in room r1 and robot in room r2, and (b) a normalized PDDL description of the Gripper task.

called *lifted*. The *initial state specification I* is a conjunction of ground atoms with fluent predicates. The *goal specification G* is a conjunction of ground literals (atoms or its negations). $\mathcal{O}$ is a finite set of *schematic operators*. A schematic operator $o = \langle head(o), pre(o), add(o), del(o) \rangle$ consists of the atom $head(o)$ indicating the name of the operator, the preconditions $pre(o)$, the add-effects $add(o)$, and the delete-effects $del(o)$, each of which is a conjunction of literals over $\mathcal{L}$.

For each operator o, the set of operator parameters $params(o)$ is defined as $free(pre(o)) \cup free(add(o)) \cup free(del(o))$. All the variables of a schematic operator are free. Operators with empty parameter sets are called *ground* operators. Otherwise, an operator can be grounded by replacing parameters with possible objects in the domain.

We use notation $o_{\downarrow}(P/a)$ to denote a set of ground operators induced by assigning objects a to parameter subset P and grounding the remaining parameters with all possible objects. For instance, in the gripper example, the ground operator set of the schematic operator o=pick(?b,?r,?g) induced by the assignment {?b/b1,?g/g1} is

$$o_{\downarrow}(\{\texttt{?b/b1,?g/g1}\}) = \{\texttt{pick(b1,r1,g1)}, \texttt{pick(b1,r2,g1)}\},$$

where the parameters ?b and ?g are replaced by the given objects but the parameter ?r is replaced with all possible room objects, {r1, r2}.

A *state s* assigns values TRUE and FALSE to all ground atoms with *fluent* predicates. The *initial state* s_0 of the task assigns value TRUE to all atoms occurring in I, and FALSE to all other fluent ground atoms. A ground operator o is *applicable* in state s if $s \models pre(o)$, that is, the preconditions of o are satisfied in the state s. A ground atom a is TRUE in the successor state if and only if either it has been TRUE in s and $a \notin del(o)$ or $a \in add(o)$. A *plan* for the task is a sequence of ground operators whose subsequent application leads from s_0 to some state s_* with $s_* \models G$.

PDDL task as MDP A Markov decision process (MDP) $M = \langle S, A, P, R \rangle$ contains a set of states S, a set of actions A, a transition probability distribution $P : S \times S \times A \mapsto [0, 1]$, and a reward function $R : S \mapsto \mathbb{R}$. When a PDDL task Π is cast as an MDP M, the states $S \in M$ is defined as the set of all states reachable from $I \in \Pi$, the action set $A \in M$ is defined as the set of labels L that is composed of a unique label for each of the ground operators, the probability distribution P is defined to respect the state-transition in the PDDL operators, and the reward function R is defined as some positive integer when $s \models G$. In practice, for each of the ground operators, the head of the ground operator $head(o)$ is assigned as the unique label.

Lifted Mutex Groups A *mutex group* is a set of mutually exclusive ground predicates M, of which at any given (reachable from I) state s at most one can be TRUE. That is $\forall s, |M \cap s| \leq 1$ or equivalently $\forall s, |\{a \mid s \models a, a \in M\}| \leq 1$. For example, in the gripper domain, {at(b1,r1),at(b1,r2)} is a mutex group as, in any given state, ball b1 can only be in one of the rooms. Any subset of a mutex group is also a mutex group. A **Lifted Mutex Group** (LMG) is a set of lifted predicates that produces a mutex group when grounded. Formally, a lifted mutex group is defined using an *invariant candidate*.

An **invariant candidate** is a tuple $c = \langle v^f(c), v^c(c), atoms(c) \rangle$ where $v^f(c)$ is a finite set of fixed variables, $v^c(c)$ is a finite set of counted variables, and $atoms(c)$ is a finite set of atoms such that all the variables of the atoms are present in either $v^f(c)$ or $v^c(c)$, i.e. $free(atoms(c)) = v^f(c) \cup v^c(c)$ and $v^f(c) \cap v^c(c) = \phi$. For example, consider an invariant candidate $c = \langle \{\texttt{?b}\}, \{\texttt{?r}\}, \{\texttt{at(?b,?r)}\} \rangle$.

Different groundings of fixed variables $v^f(c) = \{\texttt{?b}\}$ generate different sets of ground atom and different grounding of counted variable $v^c(c) = \{\texttt{?r}\}$ generates ground atoms within each set. We denote ground atom set with downarrow $_\downarrow$. One of the ground atom sets for $\{\texttt{?b/b1}\}$ is $c_\downarrow(\texttt{?b/b1}) = \{\texttt{at(b1,r1)}, \texttt{at(b1,r2)}\}$ and another ground set for $\{\texttt{?b/b2}\}$ is $c_\downarrow(\texttt{?b/b2}) = \{\texttt{at(b2,r1),at(b2,r2)}\}$.

An invariant candidate is called a **lifted mutex group** if all of its ground atom sets are mutex groups, that is, $\forall x, s, |\{a \mid s \models a, a \in c_\downarrow(v^f(c)/x)\}| \leq 1$. Here without loss of generality, we assume that each LMG has only one atom. If an LMG does not have any fixed variable, then it can generate only one ground mutex group. For example, $\langle \emptyset, \texttt{?r}, \{\texttt{at\_robby(?r)}\}\rangle$ induces only one ground atoms set $\{\texttt{at\_robby(r1),at\_robby(r2)}\}$. Fiser (2020) provides a method to identify the set of LMGs for a PDDL task. In what follows, a lifted mutex group l is called *relevant* to an operator o if $atom(l) \in pre(o)$.

Label Reduction

The paper aims to reduce the action space of an AI planning task, described as an MDP for RL. As discussed in previous section, the set of RL actions or the label set L for such an MDP consists of label $head(o)$ for each ground operator o. Here, we identify an assignment of labels to planning operators such that it generates a smaller label set L', while producing an equivalent transition system. We capture this requirement by specifying the criteria for a *valid label reduction*. A label reduction is *valid* if it assigns distinct labels to any two ground operators that can be applied in a reachable state. For example, operators `pick(b1,r1,g1)` and `pick(b2,r2,g1)` cannot be applied in the same state as the gripper `g1` cannot be in two different rooms in the same state. Thus, assigning the same label to both would be valid. But `pick(b1,r1,g1)` and `pick(b2,r1,g2)` can be applied in the same state, and hence cannot be assigned the same action label. Formally,

Definition 1 *A label reduction function $\alpha : L \mapsto L'$ is* ***valid*** *if any two distinct operator labels $head(o_1), head(o_2) \in L$ that are applicable in the same reachable state $(s \models pre(o_1) \land s \models pre(o_2))$ are assigned distinct labels, that is $\alpha(head(o_1)) \neq \alpha(head(o_2))$.*

This definition ensures that any two operators that are applicable in a same state are distinguishable. For each reduced label, the set of corresponding operators must include at most one applicable operator for each reachable state. Noticing the resemblance to predicate mutex groups, we call such operator sets *applicable operator mutex groups*.

Definition 2 *The set of ground operators $\mathcal{O}'$ is an* ***applicable operator mutex group*** *(AOMG) if for any reachable state s, $|\{o \mid s \models pre(o), o \in \mathcal{O}'\}| \leq 1$.*

Naturally, any subset of an AOMG is also an AOMG, and any subset of operators of size 1 is an AOMG. A partitioning of operators into AOMGs defines a valid label reduction, and vice versa, a valid label reduction defines a partitioning of operators into AOMGs. The problem of finding a label set L' of size $m \leq |L|$ is equivalent to the problem of finding a set cover of size m given a set of AOMGs $\mathcal{O}'_1 \ldots \mathcal{O}'_k$ such that $\bigcup_{i=1}^k \mathcal{O}'_k$ is the set of all ground operators induced by $\mathcal{O}$ and $m < k$. However, the set cover problem is defined over ground operator sets, and it becomes impractical to compute when the set of ground operators is large. Thus, we focus on finding AOMGs for (lifted) schematic operators. We find AOMGs for each schematic operator separately, by reducing its parameters. For example, given a schematic operator $o = \texttt{pick(?b,?r,?g)}$ as a robot can only be in one specific room in any state, only one specific assignment to $?r$ is satisfiable in any state. So one possible set of AOMGs can be obtained by defining partial grounding of operator o on the subset of parameters obtained after removing $?r$. That is $\{o_\downarrow(\{\texttt{?b/b,?g/g}\}) \mid \forall \texttt{b,g}\} = \{\{\texttt{pick(b1,r1,g1),pick(b1,r2,g1)}\}, \{\texttt{pick(b1,r1,g2),pick(b1,r2,g2)}\}, ...\}$.

Any (partial) parameter grounding defines a partitioning over set of (ground) operators, where each partition corresponds to a particular assignment of objects to a subset of parameters. Thus, we want to identify a *subset of parameters* (X) such that any assignment (c) to this subset results in the ground operator set $(o_\downarrow(X/c))$ being an AOMG (like the subset $\{?b, ?g\}$ in the above example). Note that LMGs have a similar property. Any assignment to their fixed variables results in a ground atom set being a mutex group. Next we show how LMGs can be used to identify the required parameter subset.

Theorem 1 *Given a schematic operator o and a lifted mutex group $l = \langle v^f(l), v^c(l), atom(l)\rangle$, if $atom(l) \in pre(o)$, then any assignment c to $X = params(o) \setminus v^c(l)$ results in $o_\downarrow(X/c)$ being an AOMG.*

Proof: Given an assignment $v^f(l)/c$, any state s can only satisfy at most one of the ground atoms from the mutex group $l_\downarrow(v^f(l)/c)$ (from the definition of LMG). Consequently, as $atom(l) \in pre(o)$, the state can satisfy at most one of the preconditions of the ground operators in the set $o_\downarrow(X/c)$. Hence, $o_\downarrow(X/c)$ is an AOMG. ∎

The parameters from set $v^c(l)$ need not be included in X, as given the assignment to $v^f(l) \subset params(o)$, the LMG l guarantees a unique assignment to parameters $v^c(l)$. Once the assignment to these parameters is identified, another LMG l' could now be used to identify the assignment to parameters $v^c(l')$ and hence $v^c(l')$ can also be removed from X. Essentially, we can leverage multiple LMGs to further reduce the subset X. Formally, this corresponds to the following problem, which we call *parameter seed set*:

Input: A schematic operator o with parameters $params(o)$ and a set of *relevant* lifted mutex groups L.
Find: A subset $X \subseteq params(o)$ of parameters s.t. $\exists X_1, \ldots X_k$ with (i) $X = X_1 \subseteq X_2 \subseteq \ldots \subseteq X_k = params(o)$, and (ii) $X_{i+1} = X_i \cup v^c(l)$ for some $l \in L$ s.t. $v^f(l) \subseteq X_i$.

Any assignment of objects to the parameter seed set X will result in a unique assignment to all the remaining parameters of o for any reachable state.

Theorem 2 *Let o be a schematic operator over parameters $params(o)$ and X be a solution to the parameter seed set problem above. Any assignment c of objects to X results in $o_{\downarrow}(X/c)$ being an AOMG.*

Proof: Let $X_1 \subseteq X_2 \subseteq \ldots \subseteq X_k = params(o)$ and let $l_1 \ldots l_{k-1}$ be lifted mutex groups such that $v^f(l_i) \in X_i$ and $X_{i+1} = X_i \cup v^c(l_i)$. Then, each X_i is a solution to the parameter seed set problem. We prove the claim by induction over the number of lifted mutex groups starting from k. Base claim of $o_{\downarrow}(X_k/x)$ (one lifted mutex group) is AOMG results from Theorem 1. Now we assume that $o_{\downarrow}(X_{i+1}/c_1)$ is an AOMG for any assignment $\hat{c}$ to X_{i+1} and prove that $o_{\downarrow}(X_i/\tilde{c})$ is an AOMG for any assignment $\tilde{c}$ to X_i. Since $l_i = \langle v^f(l_i), v^c(l_i), atom(l_i)\rangle$ is a lifted mutex group with $v^f(l_i) \subseteq X_i$, we have that $l_{i\downarrow}(X_i/\tilde{c})$ is a mutex group. Let o_1, o_2 be two ground operators in $o_{\downarrow}(X_i/\tilde{c})$. If both o_1 and o_2 belong to $o_{\downarrow}(X_{i+1}/\hat{c})$, we are done. Otherwise, assume o_1 in $o_{\downarrow}(X_{i+1}/c_1)$ and o_2 in $o_{\downarrow}(X_{i+1}/c_2)$, where c_1 and c_2 agree on X_i but differ on $X_{i+1} \setminus X_i$. However, since $X_{i+1} = X_i \cup v^c(l_i)$, we have $X_{i+1} \setminus X_i \subseteq v^c(l_i)$, making c_1 and c_2 being mutually exclusive. Thus, $o_{\downarrow}(X_i/\tilde{c})$ is an AOMG. ∎

The parameter seed set problem is at least as hard as (optimal) delete-free STRIPS planning: a reduction introduces a lifted mutex group for each STRIPS ground operators, fixed variables as preconditions and counted variables as add effects. Interestingly, going in the other direction, to solve the parameter seed set problem, we cast it as a (delete-free) STRIPS planning task. We first find a set L of relevant LMGs. Then, for each schematic operator o we define a separate planning task $\Pi_o = \langle \mathcal{L}_o, \mathcal{O}_o, I_o, G_o\rangle$, where

- Language $\mathcal{L}_o$ contains a single predicate mark and an object for each parameter in $params(o)$.
- The set of operators $\mathcal{O}_o$ consists of two schematic operators seed and get$_l$, one per each relevant LMG l, where seed $:= \langle \mathrm{seed}(x), \emptyset, \{\mathrm{mark}(x)\}, \emptyset\rangle$ and get$_l$ $:= \langle get_l, \{\mathrm{mark}(x) \mid x \in v^f(l)\}, \{\mathrm{mark}(y) \mid y \in v^c(l)\}, \emptyset\rangle$.
- Initial state $I_o = \emptyset$
- Goal state $G_o = \{\mathrm{mark}(x) \mid \forall x \in params(o)\}$.

Here, the operator seed marks each parameter $x \in params(o)$ as an element of the seed set. Operator get$_l$ indicates that a unique assignment for the parameters $x \in v^c(l)$ can be identified if all parameters $y \in v^f(l)$ are known. Therefore, the parameters $v^c(l)$ can be reduced. A plan for Π_o corresponds to a sequence of seed and get$_l$ operators. The parameters marked by seed operator form the seed set and the rest are reduced.

Theorem 3 *For a plan π of Π_o, $X_\pi = \{c \mid \mathrm{seed}(c) \in \pi\}$, is a solution to the parameter seed set problem of o.*

Proof: Let π be a plan for Π_o (assume there are no redundant repetitions of operators in π). Further, since seed operators have no preconditions, assume these operators come before get$_l$ operators, and let $\pi = \pi_s \pi_g$ denote the partition of π into the two sequences of seed and get$_l$ operators, respectively. Further, let s_1 be the state resulting from applying π_s in the initial state I_o and $s_1, \ldots, s_k$ be the sequence of states along π_g applied to s_1. Then, we have (i) $s_1 \subseteq s_2 \subseteq \ldots \subseteq s_k$ and $s_k = \{\mathrm{mark}(x) \mid x \in params(o)\}$, as well as (ii) $s_{i+1} = s_i \cup add(get_l) = \{\mathrm{mark}(y) \mid y \in v^c(l)\}$ for some get$_l$ with $pre(get_l) = \{\mathrm{mark}(x) \mid x \in v^f(l)\} \subseteq s_i$. Denoting now the parameters of o mentioned in the state s by $X(s) = \{x \mid \mathrm{mark}(x) \in s\}$, we get that $X_\pi = X(s_1)$ as requested. ∎

Different plans may correspond to different parameter seed sets X and thus different AOMGs. To find the smallest possible label set L', we want to minimize the number of AOMGs. So our objective is to find a seed set X with a minimal number of possible assignments. This objective is not linear. To overcome this issue, we generate all plans of the planning problem and choose the plan that minimizes an estimate of the possible number of assignments. This estimate for a seed set X is the product of the number of objects for each parameter in X.

To summarize, we find a parameter seed-set X for each schematic operator such that assigning objects to X will result in a set of ground operators that is an AOMG. Hence, all the ground operators in that set can be assigned the same reduced label.

Experiments

We explicitly aim at answering the following questions:

Q1: Does our approach reduce the size of the action label set? Is the reduction substantial?

Q2: Can the reduction help with learning RL policies?

To address **Q1**, we compare the size of label sets, obtained with and without the proposed reduction, on a representative set of 14 STRIPS domains from various IPC (using the typed versions where available). We use the implementation by Fiser (2020) for inferring the lifted mutex

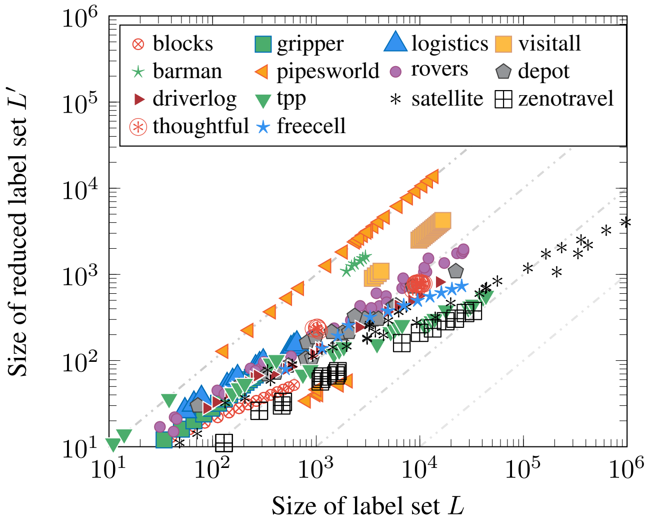


Figure 2: Comparison of label set sizes.

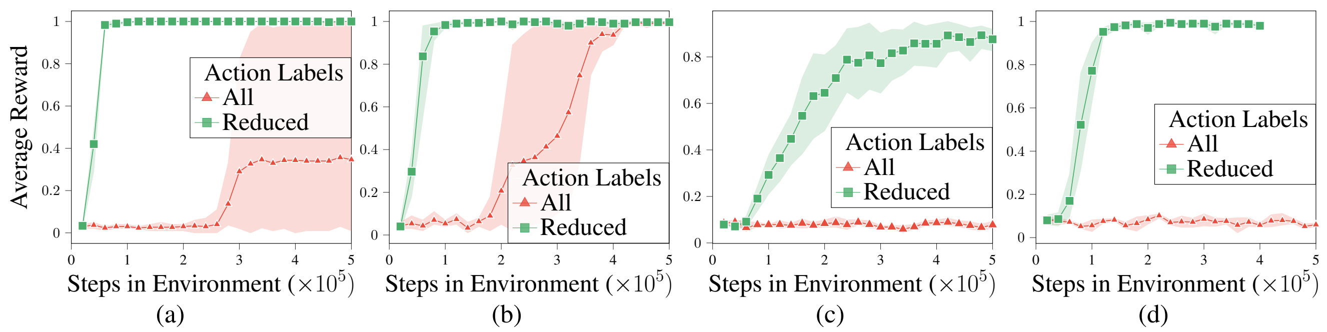


Figure 3: Learning curve in (a) ferry, (b) gripper, (c) blocks, and (d) logistics; with and without action label reduction.

groups, the ForbidIterative (FI) unordered top quality planner (Katz, Sohrabi, and Udrea 2020) to solve the *parameter seed set* planning task, and the Fast Downward planning system translator (Helmert 2006) to ground the schematic operators. As explained in the previous section, we get all the solutions to the parameter seed set planning task with FI planner (using $|params(o)|$ quality multiplier) and choose one that has the lowest estimate of possible assignments.

Figure 2 compares the size of the label sets L' and L, obtained with and without the reduction resp., for each PDDL problem instance of the 14 STRIPS domains. Both axes are log-scaled. Points below the diagonal indicate instances where the label set with proposed reduction is smaller than the original one. The distance from the diagonal indicates the significance of the reduction. Gray dashed lines below the diagonal represents the order of magnitude of the reduction. *Our experimental results show a substantial reduction of the label set in most problem instances, going beyond* 2 *orders of magnitude on some problems.*

Table 1 summarizes the number of schematic operators that were reduced by our approach in those 14 IPC domains. It also presents the mean and max number of reducible parameters found in the schematic operators, i.e., $|params(o)| - |X|$. Each row of the table represents a domain, aggregating results over the instances of that domain. Of the 14 domains considered, in 11 domains we were able to reduce at-least one parameter in all the schematic operators. The largest reduction, in terms of the number of reducible parameters, is observed in *freecell, pipesworld, thoughtful*, *tpp*, and *zenotravel* domains. Each having 3–5 parameters reduced for some schematic operators. Note that the number of reduced parameters (in Table 1) is not proportional to the number of reduced actions (in Figure 2). Nevertheless, the number of reduced parameters indicate the importance of parameter reduction. The total compute time did not exceed 4 seconds for any of these problems.

Moving on to question **Q2**, to evaluate the advantage of reducing the label set size, we cast the PDDL task as an MDP with the reduced label set and learn an RL policy. We focus on 4 classical planning domains from Rivlin, Hazan, and Karpas (2020): *Ferry, Gripper, Blocks,* and *Logistics*. Since our aim is to evaluate the advantage of reducing the action space, and not to evaluate the generalization of policies, we fixed the number of objects in each domain. We

Domain	# reduced operators	reducible parameters	
		max % (#)	mean % (#)
blocks	3/4	50% (0.75)	50% (0.75)
gripper	3/3	49% (1.33)	49% (1.33)
logistics	6/6	58% (1.83)	55% (1.76)
visitall	1/1	50% (1.00)	50% (1.00)
barman	10/12	42% (2.00)	42% (2.00)
pipesworld	6/6	79% (5.00)	60% (3.87)
rovers	9/9	63% (2.40)	54% (2.10)
depot	5/5	47% (1.80)	47% (1.80)
driverlog	6/6	47% (1.50)	47% (1.50)
tpp	4/4	62% (4.00)	62% (4.00)
satellite	5/5	93% (2.60)	52% (1.46)
zenotravel	5/5	79% (3.40)	62% (2.68)
thoughtful	20/21	73% (3.24)	73% (3.24)
freecell	10/10	65% (3.30)	65% (3.30)

Table 1: Summary of lifted operators reduced by our approach in 14 IPC domains. Column 2 presents the number of lifted operators that were reduced out of total number of lifted operator in the domain file. Column 3 & 4 presents the maximum & mean of percent (number) of reducible parameters found per operator, i.e $|params(o)| - |X|$.

generate 500 unique pairs of initial and goal state in each domain. Of these, 250 were used in training and remaining were set aside for evaluation. Table 2 summarizes the number of objects selected in each domain and the number of action labels. We used PDDLEnv[1] library to convert the PDDL domain and problem files to RL Environment. Inspired by the work of Gehring et al. (2022), we used a domain-independent planning heuristic, h^{FF}, as dense reward function. We employed the Double DQN (van Hasselt, Guez, and Silver 2016) implementation from the ACME RL library (Hoffman et al. 2020) to learn a state-action value function and applied a greedy policy $\pi(s) = \max_a Q(s, a)$

[1]https://github.com/gehring/pddlenv

Domain	Objects	# Action Labels	
		Grounded	Reduced
Blocks	4 blocks	40	13
Ferry	3 cars 3 locations	24	7
Gripper	4 balls 2 locations	36	14
Logistics	2 packages, 2 cities, 2 trucks, 1 airplane	68	20

Table 2: Summary of domains used in RL evaluations.

Hyperparameters	**Values**	**Exception**
Learning Rate	0.003	–
Batch Size	4	16 in Logistics
Input Size	# possible state literals	
Output Size	# Action labels	
Hidden layers	3	–
Hidden units	64	512 in Logistics
Discount	0.95	–
Max Episode Length	100	–

Table 3: Summary of hyperparameters used in RL evaluations.

in our evaluation. Table 3 describes all the hyperparameters used in the domains. The experiments were performed on computing clusters with Intel(R) Xeon(R) CPU E5-2667 v2 @ 3.30GHz and Tesla K80.

Figure 3 shows learning curves aggregated over 5 runs with random seeds. For Ferry and Gripper domains (Figure 3 a and b), the reduction of action labels improves the sample efficiency by as many as $300,000$ steps. In Blocks and Logistics domains (Figure 3 c and d), the baseline without the label reduction was not able to learn a policy. Once a reduced label set is used, the training becomes feasible. It is clear from these plots that reducing the action label set yields significant gain in terms of sample efficiency. The problem files that are used for the RL evaluation can be considered minuscule, since there are only a few objects (4–7) in each of the domains and there are not too many labels to begin with. However, our results show that even in such small-scale problems reducing the action labels is advantageous. In large problems (with many objects) our approach can provide tremendous leverage for training RL algorithms as action set is reduced by order of magnitudes. With this, we answer the **Q2** affirmatively.

Related Work

Various approaches have been studies in RL to reduce the action space. Stochastic action sets (Boutilier et al. 2018) and invalid action masking (Huang and Ontañón 2020; Bamford and Ovalle 2021; Kanervisto, Scheller, and Hautamäki 2020) restricts the action selected by an agent to a small subset of actions that are feasible in the given state. This is done by assigning zero probability (or -inf score) to invalid actions. While the stochastic action sets and invalid action masking define a state-dependent subset of feasible actions, our action reduction is independent of the current state. Indeed, our approach can be combined with invalid action masking to further accelerate the convergence.

Another approach to manage large number of actions in an MDP is by using factored action spaces (Pazis and Lagoudakis 2011; Geißer, Speck, and Keller 2020; Guestrin, Lagoudakis, and Parr 2002). With factored action space, an action is decomposed into multiple components and represented as either a decision tree or a vector. It is straight forward to convert predicate action space (for example, gripper actions $\{$`drop(b1, r2, g1), pick(b2, r1, g2),...`$\}$) to a factored action space $(a_0, a_1, \ldots, a_n)$ with a_0 denoting the action identifier (for example, `drop` or `pick`) and $a_1, \ldots, a_n$, denoting the parameters. Our approach of identifying the parameter seed set can be used to reduce the number of factors in the factored action spaces.

In planning literature, label reduction techniques are used to reduce the number of transition labels in an abstract transition graph (Helmert et al. 2014; Sievers, Wehrle, and Helmert 2014). The aim of label reduction in this setting is to simplify the transition system by creating equivalence between labelled transitions. In this work, the purpose of label reduction is different. Labels of operators that are never applicable together are reduced to the same label, allowing to differentiate applicable operators in a given state.

Discussion and Future Work

In this work, we have introduced definitions of a valid label reduction and applicable operator mutex groups and have shown the connection between the two. We have presented a method for automatically deriving operator label reductions for planning tasks based on operator parameter reduction. For that, a parameter seed set problem was introduced, and a solution to the problem was suggested by translating it to delete-free planning. Our experimental evaluation shows a significant reduction in operator labels when using our approach, across all tested planning domains. This reduction translates into improved performance of standard RL agents on the tested problems.

Our method, however, does not guarantee optimality of the valid reduction size, even for the restricted case considered in this work. Finding provably minimal size reductions is an interesting topic for future research. Further, we have not explored the possible benefits of operator parameter reduction for lifted planning, such as for faster successor generation (Corrêa et al. 2020) or heuristic computation (Corrêa et al. 2021; Lauer et al. 2021).

References

Bamford, C.; and Ovalle, A. 2021. Generalising Discrete Action Spaces with Conditional Action Trees. In *2021 IEEE Conference on Games (CoG)*, 1–8.

Boutilier, C.; Cohen, A.; Hassidim, A.; Mansour, Y.; Meshi, O.; Mladenov, M.; and Schuurmans, D. 2018. Planning and Learning with Stochastic Action Sets. In *IJCAI*, 4674–4682.

Corrêa, A. B.; Francès, G.; Pommerening, F.; and Helmert, M. 2021. Delete-Relaxation Heuristics for Lifted Classical Planning. In *ICAPS*, 94–102.

Corrêa, A. B.; Pommerening, F.; Helmert, M.; and Francès, G. 2020. Lifted Successor Generation using Query Optimization Techniques. In *ICAPS*, 80–89.

Dzeroski, S.; Raedt, L. D.; and Driessens, K. 2001. Relational Reinforcement Learning. *ML*, 43(1/2): 7–52.

Fiser, D. 2020. Lifted Fact-Alternating Mutex Groups and Pruned Grounding of Classical Planning Problems. In *AAAI*, 9835–9842.

Gehring, C.; Asai, M.; Chitnis, R.; Silver, T.; Kaelbling, L. P.; Sohrabi, S.; and Katz, M. 2022. Reinforcement Learning for Classical Planning: Viewing Heuristics as Dense Reward Generators. In *ICAPS*.

Geißer, F.; Speck, D.; and Keller, T. 2020. Trial-Based Heuristic Tree Search for MDPs with Factored Action Spaces. In *SOCS*, 38–47.

Guestrin, C.; Lagoudakis, M. G.; and Parr, R. 2002. Coordinated Reinforcement Learning. In *ICML*, 227–234.

Helmert, M. 2006. The Fast Downward Planning System. *JAIR*, 26: 191–246.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces. *Journal of the ACM*, 61(3): 16:1–63.

Hoffman, M.; Shahriari, B.; Aslanides, J.; Barth-Maron, G.; et al. 2020. Acme: A Research Framework for Distributed Reinforcement Learning. *CoRR*, abs/2006.00979.

Huang, S.; and Ontañón, S. 2020. A closer look at invalid action masking in policy gradient algorithms. *arXiv preprint arXiv:2006.14171*.

Kanervisto, A.; Scheller, C.; and Hautamäki, V. 2020. Action Space Shaping in Deep Reinforcement Learning. In *2020 IEEE Conference on Games (CoG)*, 479–486.

Katz, M.; Sohrabi, S.; and Udrea, O. 2020. Top-Quality Planning: Finding Practically Useful Sets of Best Plans. In *AAAI*, 9900–9907.

Lauer, P.; Torralba, Á.; Fišer, D.; Höller, D.; Wichlacz, J.; and Hoffmann, J. 2021. Polynomial-Time in PDDL Input Size: Making the Delete Relaxation Feasible for Lifted Planning. In *IJCAI*, 4119–4126.

McDermott, D. V. 2000. The 1998 AI Planning Systems Competition. *AI Mag.*, 21(2): 35–55.

Pazis, J.; and Lagoudakis, M. G. 2011. Reinforcement learning in multidimensional continuous action spaces. In *IEEE Symposium on Adaptive Dynamic Programming And Reinforcement*, 97–104.

Pazis, J.; and Parr, R. 2011. Generalized Value Functions for Large Action Sets. In *ICML*, 1185–1192.

Rivlin, O.; Hazan, T.; and Karpas, E. 2020. Generalized Planning With Deep Reinforcement Learning. *CoRR*, abs/2005.02305.

Röger, G.; Sievers, S.; and Katz, M. 2018. Symmetry-based Task Reduction for Relaxed Reachability Analysis. In *ICAPS*, 208–217.

Sievers, S.; Wehrle, M.; and Helmert, M. 2014. Generalized Label Reduction for Merge-and-Shrink Heuristics. In *AAAI*, 2358–2366.

Silver, T.; and Chitnis, R. 2020. PDDLGym: Gym Environments from PDDL Problems. *CoRR*, abs/2002.06432.

van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep Reinforcement Learning with Double Q-Learning. In *AAAI*, 2094–2100.

Zahavy, T.; Haroush, M.; Merlis, N.; Mankowitz, D. J.; and Mannor, S. 2018. Learn What Not to Learn: Action Elimination with Deep Reinforcement Learning. In *NeurIPS 2018*, 3566–3577.