Structural Symmetries for Fully Observable Nondeterministic Planning

Dominik Winterer
University of Freiburg
Germany
wintered@cs.uni-freiburg.de

Martin Wehrle
University of Basel
Switzerland
martin.wehrle@unibas.ch

Michael Katz
IBM Watson Health
Israel
katzm@il.ibm.com

Abstract

Symmetry reduction has significantly contributed to the success of classical planning as heuristic search. However, it is an open question if symmetry reduction techniques can be lifted to fully observable nondeterministic (FOND) planning. We generalize the concepts of structural symmetries and symmetry reduction to FOND planning and specifically to the LAO* algorithm. Our base implementation of LAO* in the Fast Downward planner is competitive with the LAO*-based FOND planner myND. Our experiments further show that symmetry reduction can yield strong performance gains compared to our base implementation of LAO*.

1 Introduction

Fully observable nondeterministic (FOND) planning is a challenging task because the application of actions can have several possible outcomes, and planning algorithms must consider all these possibilities. Solutions to FOND planning tasks are represented as policies, which map states to actions, and thus determine the next action in the final plan depending on the actual outcome of the previously applied action. The current state of the art in FOND planning includes the *Planner for Relevant Policies* [Muise *et al.*, 2012; 2014], which performs multiple runs of an underlying classical planner, and specifically focuses on relevant state parts. An alternative approach to solve FOND planning tasks is based on heuristic search with the LAO* algorithm [Hansen and Zilberstein, 2001; Mattmüller *et al.*, 2010]. Similar to solving classical planning tasks with A*, an efficient FOND planning algorithm based on LAO* features accurate distance heuristics to guide the search towards goal states, as well as (completeness-preserving) pruning methods to reduce the branching factor of the search space.

Pruning methods have shown their potential for classical planning and state-space search in recent years, and several methods have been proposed [Fox and Long, 1999; 2002; Coles and Coles, 2010; Pochter *et al.*, 2011; Wehrle and Helmert, 2012; Domshlak *et al.*, 2012; Nissim *et al.*, 2012; Wehrle *et al.*, 2013; Domshlak *et al.*, 2013; Wehrle and Helmert, 2014; Holte and Burch, 2014; Holte *et al.*, 2015]. In particular, *symmetry reduction* in the form of orbit space search has demonstrated to be effectively applicable in a wide range of classical planning domains, often reducing the state space size and increasing planning performance significantly [Domshlak *et al.*, 2012; 2015]. In a nutshell, for classical planning, symmetry reduction computes equivalence classes of "symmetrical" states, with the property that states in equivalence classes need not be distinguished during the search because they in turn yield equivalent "symmetrical" paths in the state space. For example, assume that two objects O_1 and O_2 have to be carried with a vehicle V, where V has a right and a left place to load these objects. Then the state where O_1 is loaded on the left and O_2 on the right, and the state where O_1 and O_2 are loaded on the opposite positions, are symmetrical. Informally, the actual object position of O_1 and O_2 on the vehicle does not matter to carry them to the goal position, and hence, only one of these states needs to be considered.

While symmetry reduction has shown to be beneficial for A* in "classical" state spaces, it is an open question whether it can be effectively generalized to FOND planning with LAO*. We provide the framework for such a generalization, in particular addressing the questions how the concept of symmetries can be generalized to operators with nondeterministic effects, how the well-established orbit space search algorithm from classical planning can be generalized to AND/OR graphs, and how concrete policies can be reconstructed from "symmetrical" policies. We have implemented our framework within the Fast Downward planning system [Helmert, 2006], and evaluated our approach on a variety of FOND planning benchmarks. Our experiments show the potential of symmetries for LAO*, both compared to the myND planner [Mattmüller *et al.*, 2010] and compared to vanilla LAO*. As a side result, our implementation offers a competitive LAO*-based FOND planner to the community, making Fast Downward's features applicable to FOND planning as well.

2 Preliminaries

We use an SAS$^+$ based notation [Bäckström and Nebel, 1995] to model fully observable nondeterministic planning tasks with a finite set of finite-domain state variables $\mathcal{V}$. Every variable v in $\mathcal{V}$ has a finite domain $dom(v)$. A variable/value pair $\langle v,d\rangle$ for $v \in \mathcal{V}$ and $d \in dom(v)$ is called a *fact*. A *partial state* s is a function from variables $\mathcal{V}_s \subseteq \mathcal{V}$ to values in the domains of $\mathcal{V}_s$, whereas all variables in $\mathcal{V} \setminus \mathcal{V}_s$ have an undefined value $\perp$. We denote the value of v in s with $s[v]$

(including $s[v] = \bot$ if $v \in \mathcal{V} \setminus \mathcal{V}_s$). By $vars(s)$ we denote the subset of variables for which a value is defined by the partial state s. A *state* is a partial state where all values are defined, i.e., with $vars(s) = \mathcal{V}$. A state s complies with a partial state t, denoted by $s \supseteq t$, if $s[v] = t[v]$ for all $v \in vars(t)$.

We consider *fully observable nondeterministic* (FOND) planning tasks. In contrast to classical planning, operators in FOND planning can have several possible effects, where only one of these effects is applied when the operator is executed. Formally, FOND planning tasks are defined as follows.

Definition 1 (fully observable nondeterministic planning task)**.** *A fully observable nondeterministic (FOND) planning task is a tuple $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$, where*

- *$\mathcal{V}$ is a finite set of finite-domain state variables,*
- *$\mathcal{O}$ is a finite set of nondeterministic operators,*
- *s_0 is the initial state, and*
- *$s_\star$ is a partial state called goal.*

A nondeterministic operator *$o \in \mathcal{O}$ has the form $\langle pre(o), effs(o) \rangle$, where $pre(o)$ is a partial state that denotes the* precondition *and $effs(o)$ is the set of* effects *of o, where each effect in $effs(o)$ is a partial state.*

In the following, we will denote a *nondeterministic operator* as an *operator* unless stated otherwise. An operator o is *applicable* in a state s iff $s[v] = pre(o)[v]$ for all $v \in vars(pre(o))$. If operator o is applicable in s, the set of *successor states* $o(s) := \{o_{eff}(s) \mid eff \in effs(o)\}$ of s is obtained from s based on the successor states $o_{eff}(s)$ for each $eff \in effs(o)$, where $o_{eff}(s)$ is obtained from s by setting the values of variables in $vars(eff)$ to their values in eff, and leaving the remaining variable values unchanged. We denote the set of applicable operators in s by $app(s)$.

For a FOND planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$, we define the *transition system* of Π as $\mathcal{T}^\Pi = \langle S, T, s_0, S_\star \rangle$, where S is the set of Π's states, and $T \subseteq S \times \mathcal{O} \times S$ is the set of transitions, with $\langle s, o, s' \rangle \in T$ iff o is applicable in s, and s' is a possible outcome of applying o in s, i.e., $s' \in o(s)$. Furthermore, s_0 is the initial state, and $S_\star$ is the set of goal states, i.e., the set of states that comply with $s_\star$.

As mentioned, operators generally can have several effects, and applying an operator in a state yields a successor state where (nondeterministically) one of these effects has been applied. Hence, solutions to FOND planning tasks are not linear operator sequences as in classical planning, but rather *policies* which determine the next plan operator depending on the outcome of the previously applied plan operator.

Definition 2 (policy)**.** *Let Π be a FOND planning task with set of states S. A policy is a mapping $\pi : S \to \mathcal{O} \cup \{\bot\}$, which maps states to operators, or π is undefined (i.e., $\pi(s) = \bot$).*

We shortly denote the sequential application of the operators determined by π as *following* π. A policy π is called *weak* if π defines at least one path from the initial state to a goal state when following π. In this case, π is called a *weak plan* for Π. A policy π is *closed* if following π either leads to a goal state, or to a state where the policy is defined. It is *proper* if from every state visited following π there exits a path to a goal state following π. A policy that is closed and proper is called a *strong cyclic plan* for Π. Furthermore, π is *acyclic* if it does not revisit already visited states. A closed and proper acyclic policy is called a *strong plan* for Π.

Informally, a weak plan is a sequence of operators which leads to a goal state if all nondeterministic operator outcomes were deterministic, which corresponds to a plan in classical planning. A strong plan guarantees that a goal state is reached, where an upper bound on the number of plan steps exists. In contrast, strong cyclic plans require a finite number of steps to reach a goal state, but no such upper bound on the plan steps can be provided a priori. However, strong cyclic planning is based on the fairness assumption, i.e. there is a nonzero probabilty that a goal state can be reached when following a strong cylic plan. We focus on the problem of finding strong cyclic plans with the LAO* algorithm.

2.1 Symmetry Elimination in Classical Planning

Symmetry elimination has shown its potential for planning in several contexts [Fox and Long, 1999; 2002; Rintanen, 2003; Coles and Coles, 2010; Pochter *et al.*, 2011; Domshlak *et al.*, 2012; 2013; Shleyfman *et al.*, 2015]. Symmetry elimination considers equivalence classes of symmetrical states, and allows for using representative states of each equivalence class. Recently, Shleyfman et al. [2015] introduced the notion of *structural symmetries*, which captures previously proposed concepts of symmetries for classical planning. In a nutshell, structural symmetries directly work on the factored representation of a classical planning task Π. They map operators to operators, and facts to facts in a way that the induced mapping on the state space graph $\mathcal{T}_\Pi$ is an automorphism of $\mathcal{T}_\Pi$.

Symmetry elimination has been considered in particular for planning as heuristic search. For a classical planning task Π with state set S, a set of structural symmetries Σ induces a group Γ and an equivalence relation $\sim_\Gamma$ on S, where $s \sim_\Gamma s'$ iff there is $\sigma \in \Gamma$ such that $\sigma(s) = s'$. For a state s, pruning algorithms based on symmetry elimination only consider the equivalence classes of the successor states of s instead of all successor states, and only keep one representative element of these classes. In this sense, A* with symmetry elimination applies all operators in s, but prunes some of the resulting successor states. The resulting reduced state transition graph is guaranteed to still contain an optimal plan in s. To achieve this, an algorithm called *orbit space search* has been introduced [Domshlak *et al.*, 2015]. Orbit space search works directly in the *orbit space* induced by canonical representatives: Orbit space search searches the state space graph induced by canonical representatives for each encountered state, where states are mapped to corresponding representatives. As computing single canonical states for all symmetrical states is not tractable, a greedy approach is employed to compute symmetrical states in a safe way.

3 Structural Symmetries for FOND Planning

Structural symmetries [Shleyfman *et al.*, 2015] are naturally extended to FOND planning tasks.

Definition 3 (structural symmetry)**.** *For a FOND planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$, let F be the set of Π's facts, i.e.,*

pairs $\langle v, d\rangle$ *with* $v \in \mathcal{V}$, $d \in dom(v)$. *A* structural symmetry *for* Π *is a permutation* $\sigma : \mathcal{V} \cup F \cup \mathcal{O} \to \mathcal{V} \cup F \cup \mathcal{O}$, *where*

1. $\sigma(\mathcal{V}) = \mathcal{V}$ *and* $\sigma(F) = F$ *such that* $\sigma(\langle v, d\rangle) = \langle v', d'\rangle$ *iff* $v' = \sigma(v)$, $d \in dom(v)$, *and* $d' \in dom(v')$
2. $\sigma(\mathcal{O}) = \mathcal{O}$ *such that for* $o \in \mathcal{O}$, $\sigma(pre(o)) = pre(\sigma(o))$, $\sigma(\mathit{effs}(o)) = \mathit{effs}(\sigma(o))$
3. $\sigma(s_\star) = s_\star$,

where $\sigma(\{x_1, \ldots, x_n\}) := \{\sigma(x_1), \ldots, \sigma(x_n)\}$, *and for a partial state* s, $s' := \sigma(s)$ *is the partial state obtained from* s *such that for all* $\langle v, d\rangle$ *with* $v \in vars(s)$ *and* $d \in dom(v)$, $\sigma(\langle v, d\rangle) = \langle v', d'\rangle$ *and* $s'[v'] = d'$.

The only difference to structural symmetries for classical planning tasks is bullet point 2., second part, where we require $\sigma(\mathit{effs}(o)) = \mathit{effs}(\sigma(o))$, i.e., several effects of o are taken into account by σ, and the set of o's effects is mapped to the corresponding effect set of o's symmetrical counterpart.

To describe how to handle symmetries, we need the notion of *Canonical mappings*.

Definition 4 (canonical mapping)**.** *A mapping* $C : S \mapsto S$ *is called a* canonical mapping *if for each state* $s \in S$ *there exists a structural symmetry* σ_s *such that* $C(s) = \sigma_s(s)$. *The state* $C(s)$ *is called the* canonical state *of* s. *A canonical mapping* C *is* perfect *if for each pair of symmetrical states* s *and* t, *we have* $C(s) = C(t)$.

As computing perfect canonical mappings is intractable, in practice, a greedy approach to compute canonical mappings is usually employed. Such a greedy approach is safe in the sense that states might not be recognized as symmetrical although they are. From now on, we assume a greedy algorithm for computing canonical mappings C. For a canonical mapping C, let $S^C = \{C(s) \mid s \in S\}$ be the set of canonical states of Π. We can now define induced *canonical transition systems*.

Definition 5 (canonical transition system)**.** *Given a transition system* $\mathcal{T}^\Pi = \langle S, T, s_0, S_\star\rangle$ *of the planning task* Π, *a canonical mapping* C *induces the* canonical transition system $\mathcal{T}^{\Pi,C} = \langle S^C, T^C, s_0^C, S_\star^C\rangle$, *where* S^C *is the set of canonical states of* Π, *and* $T^C = \{\langle C(s), o', C(t)\rangle \mid \langle s, o, \cdot\rangle \in T, o' = \sigma_s(o), t \in o'(C(s))\}$ *be the set of canonical transitions. Furthermore,* $s_0^C = C(s_0)$, *and* $S_\star^C = \{C(s) \mid s \in S_\star\}$.

For brevity, we will often write $\mathcal{T}^C$ instead of $\mathcal{T}^{\Pi,C}$ if the context is clear. In $\mathcal{T}^C$, each transition $\langle C(s), o', C(t)\rangle$ in T^C is a triplet consisting of a canonical state $C(s)$, an operator o' applicable in $C(s)$, and a canonical state $C(t)$, which is the canonical state of a state resulting from applying o' in $C(s)$. A mapping $\pi_c : S^C \mapsto \mathcal{O} \cup \{\bot\}$ is a canonical policy for Π. A canonical policy is a *weak canonical plan* if it is a weak plan for the canonical transition system, it is a *strong cyclic canonical plan* if it is a strong cyclic plan for the canonical transition system, and it is a *strong canonical plan* if it is a strong plan for the canonical transition system.

In the following, we provide an extension of the LAO* algorithm with symmetry reduction, which results in an LAO* search in a canonical transition system.

4 LAO* in Canonical Transition Systems

We propose a FOND planning algorithm with symmetry reduction, consisting of three steps: Firstly, given the FOND planning task, compute a set of structural symmetries, which in turn determines the canonical transition system. Secondly, we propose a (straight-forward) extension of the LAO* algorithm as previously used for FOND planning by Mattmüller et al. [2010]. The outcome is a strong cyclic canonical plan. Thirdly, in a post-processing phase and based on the strong cyclic canonical plan, a strong cyclic plan can be extracted.

The computation of structural symmetries is performed analogously as in classical planning. We refer the reader to the literature for details [Pochter *et al.*, 2011; Domshlak *et al.*, 2012; Shleyfman *et al.*, 2015]. For the search part, we propose a variant of LAO* [Mattmüller *et al.*, 2010], which is extended to perform its search in a canonical transition system instead of the original. This is achieved by replacing encountered states with canonical representatives. We call the resulting algorithm *canonical* LAO* (CLAO* for short). It is shown in Algorithm 1.

Algorithm 1 The CLAO* algorithm

1: Implicit graph $\mathcal{G}'$ consists of canonical initial state $C(s_0)$
2: **while** $C(s_0)$ unsolved **do**
3: $\quad E \leftarrow$ UNEXPANDEDNONGOAL(TRACE($\mathcal{G}'$))
4: $\quad$ **if** $E = \emptyset$ **then** $E \leftarrow$ UNEXPANDEDNONGOAL($\mathcal{G}'$)
5: $\quad N \leftarrow$ EXPANDALL(E)
6: $\quad$ **for each** new state $s \in N$ **do**
7: $\quad\quad$ add canonical state $C(s)$ to implicit graph $\mathcal{G}'$
8: $\quad\quad$ connect $C(s)$ to its ancestor
9: $\quad\quad f(C(s)) \leftarrow \begin{cases} 0 & \text{if } C(s) \supseteq s_\star \\ h(C(s)) & \text{otherwise} \end{cases}$
10: $\quad \mathcal{Z} \leftarrow$ BACKWARDREACH(E)
11: $\quad$ VALUEITERATION($\mathcal{Z}$)
12: $\quad$ SOLVELABELLING($\mathcal{G}'$)
13: **return** POLICYEXTRACTION(TRACE($\mathcal{G}'$))

CLAO* starts off with the canonical representative $C(s_0)$ of the initial state s_0 (line 1). As long as the algorithm does not find a solution for the canonical initial state $C(s_0)$, it performs regular LAO* for FOND planning (using a given heuristic h), with the exception that the successors resulting from operator applications are replaced by their respective canonical representatives.

As a final step, if a solution has been found by CLAO*, a concrete policy is extracted. For this step, we refer back to the definition of canonical mappings (Def. 4): For a perfect canonical mapping, it suffices to define the concrete policy as

$$\pi(s) := \sigma_s^{-1}(\pi_c(C(s)))$$

for all states s and corresponding canonical states $C(s)$. As mentioned earlier, computing perfect canonical mappings is not tractable, and hence canonical mappings will rarely be perfect in a practical setting. Consequently, if the canonical mapping is not perfect, there can be two symmetrical states, such that the canonical policy is defined for only one of them, hence the policy extraction algorithm becomes slightly

Algorithm 2 Solution reconstruction algorithm

```
1: function PolicyExtraction(π_c)
2:     Input: A canonical policy π_c
3:     OPEN is a queue initialized with s_0
4:     CLOSED is a closed list, intially empty
5:     ρ_{s_0} ← σ_{s_0}
6:     π(s_0) ← ρ_{s_0}^{-1}(π_c(C(s_0)))
7:     while OPEN is not empty do
8:         s ← pop(OPEN)
9:         Add s to CLOSED
10:        s^c ← ρ_s(s)
11:        if π_c(s^c) = ⊥ then continue
12:        o ← π(s)
13:        for each successor state t in o(s) do
14:            if t ∈ CLOSED then continue
15:            t̄ ← ρ_s(t),   t^c ← C(t̄)
16:            ρ_t ← σ_t̄ ∘ ρ_s
17:            π(t) ← ρ_t^{-1}(π_c(t^c))
18:            Add t to OPEN
19:    return policy π
```

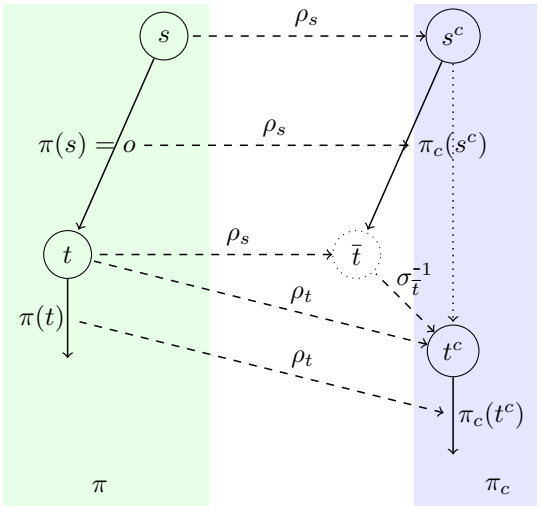

Figure 1: A step of the policy reconstruction algorithm. Dashed edges correspond to symmetries. ρ_s and $\pi(s)$ are known at the start of the step, ρ_t and $\pi(t)$ are computed.

more technical. For clarity of presentation, assuming non-perfect canonical mapping, through the section we will use the following notation. For a state s, with the canonical state $C(s)$, by s^c we denote the representative of s in the canonical policy. As mentioned above, although s, $C(s)$, and s^c are all symmetrical, s^c can be different from the canonical state $C(s)$ of s. The mapping ρ_s then denotes the structural symmetry mapping from the state s to the representative state s^c, i.e., $\rho_s(s) = s^c$.

The procedure POLICYEXTRACTION – to extract concrete policies from canonical mappings that are not perfect – is shown in Algorithm 2. It follows all possible outcomes from applying the policy, mapping the current state to a representative in the canonical policy, not necessarily its canonical state. The overall algorithm is complete.

Theorem 1. *The* CLAO* *algorithm followed by* POLICYEXTRACTION *preserves the completeness of* LAO*.

Proof. The algorithm described in Algorithm 1 is an LAO* search in the canonical transition system and thus returns a strong cyclic canonical plan, if the task is solvable. Otherwise, it returns a policy π_c such that $\pi_c(C(s_0)) = \perp$ and thus Algorithm 2 will return a policy π with $\pi(s_0) = \perp$.

Assuming that the task is solvable, we need to show that given the strong cyclic canonical plan π_c, the reconstruction step depicted in Algorithm 2 then returns a strong cyclic plan π. We show that by induction on the distance of a state from the initial state s_0, showing that for each state t with a parent s and an achieving operator $o = \pi(s)$, a corresponding state in the strong cyclic canonical plan with a policy operator defined can be found via the mapping ρ. First, for the initial state itself, its corresponding state is $C(s_0)$, mapped from s_0 via the structural symmetry $\rho_{s_0} := \sigma_{s_0}$, with $\pi_c(C(s_0))$ being defined, and the policy $\pi(s_0)$ is then obtained from $\pi_c(C(s_0))$, as in lines 5–6 of Algorithm 2.

For the induction step, assume that (i) the corresponding state $s^c = \rho_s(s)$ in π_c is defined for the parent s via ρ_s, and (ii) the policy π is defined for s, with $o = \pi(s)$ being the symmetrical operator to the operator $\pi_c(s^c)$ via ρ_s, that is $\rho_s(o) = \pi_c(s^c)$. The step is visualized in Figure 1. Let t be a state resulting from an application of o in s, and let $\bar{t} = \rho_s(t)$ be the state resulting from mapping t with its parent symmetry ρ_s (line 15). Then, $\bar{t}$ is a state resulting from an application of $\pi_c(s^c)$ in s^c. Let $C(\bar{t})$ be the canonical state of $\bar{t}$. Then, $(s^c, \pi_c(s^c), C(\bar{t}))$ is a transition in the canonical transition system that corresponds to an outcome of applying $\pi_c(s^c)$ in s^c. Thus, π_c is defined on $t^c := C(\bar{t})$. The symmetry ρ_s maps from t to $\bar{t}$ and $\sigma_{\bar{t}}$ maps from $\bar{t}$ to t^c. Thus, $\rho_t := \sigma_{\bar{t}} \circ \rho_s$ (line 16) maps from t to t^c, a state in the strong cyclic canonical plan with a policy operator defined. We can now define $\pi(t)$ by a mapping of the operator $\pi_c(t^c)$ with the inverse symmetry ρ_t^{-1} (line 17), finishing the induction step. □

We remark that in practice, there is no need for reconstructing the policy: Executing a policy can be done with the help of the canonical policy found by the CLAO* algorithm, by restricting the line 13 in Algorithm 2 to the successor t that is the actual outcome of applying the operator o in the state s.

5 Experimental Evaluation

We have implemented the variant of LAO* which has been previously used for FOND planning [Mattmüller *et al.*, 2010] as well as CLAO* within the Fast Downward planning system [Helmert, 2006]. In this section, we empirically investigate the usefulness of structural symmetries and symmetry reduction for FOND planning. In particular, we investigate the following research questions: In how many of the commonly available FOND planning domains do symmetries exist? What is the computational overhead to compute these symmetries? How much can the state space size be reduced with CLAO* compared to LAO*? And finally, does the reduction pay off in overall runtime and in the number of tasks

that can be solved? As a basis for these investigations, we compare the performance of our LAO* implementation to the LAO*-based FOND planner myND.

Our experiments have been conducted on a server equipped with 2.3 GHz Quad-Core AMD Opteron 2356 CPUs. The time and memory bounds per run are 30 minutes and 2GB, respectively. For the evaluation, we use the FF heuristic [Hoffmann and Nebel, 2001], a state-of-the-art heuristic for classical planning already available in Fast Downward (for FOND planning tasks, the FF heuristic is computed based of all-outcomes determinizations). The experiments focus on the computationally challenging parts, i.e., computing the structural symmetries, and searching with LAO* and CLAO*, excluding the policy extraction step (the latter could be done in linear time in the policy size). Our benchmark set consists of all IPC-08 FOND planning domains, including the larger tasks generated by Muise *et al.* [2012].

Fast Downward LAO* compared to myND

We first investigated the performance of our LAO* implementation in Fast Downward (shortly called FD-LAO* in the following) compared to the FOND planner myND. For both FD-LAO* and myND, we used the FF heuristic, which is available in both solvers. In Figure 2, we report the *coverage*, i.e., the number of planning tasks for which a solution has been found within our resource bounds. The total number of planning tasks per domain in reported in parenthesis.

	Coverage	
Domain	myND	FD-LAO*
BLOCKSWORLD(30)	**26**	–3
BLOCKSWORLD-2(15)	**9**	–1
ELEVATORS(15)	11	**+3**
EX-BLOCKSWORLD(15)	**8**	**±0**
FAULTS-NEW(190)	114	**+53**
FAULTS(55)	**55**	–3
FIRST-RESPONDERS(100)	83	**+14**
FOREST(100)	6	**+2**
FOREST-NEW(100)	**15**	–8
TRIANGLE-TIREWORLD(40)	**7**	**±0**
Sum (660)	334	**+57**

Figure 2: Coverage results of myND vs. FD-LAO*.

We observe that FD-LAO* is competitive with myND, and additionally, FD-LAO* has distinct advantages in the domains FAULTS-NEW and FIRST-RESPONDERS. Overall, FD-LAO* solves 57 more planning tasks, which is remarkable due to the usual exponential complexity increase in the task size. As a main conclusion, we observe that FD-LAO* provides a competitive basis for the following experiments.

CLAO* compared to FD-LAO*

In the following, we evaluate the potential of symmetry reduction for LAO*-based FOND planning. Firstly, we observed that symmetries often occur in the common FOND planning benchmarks: out of 660 planning tasks that we have considered in total, symmetries have been found in 535 of these. The preprocessing time to compute the symmetries is depicted in Figure 3. The plot shows the number of tasks for which the symmetry precomputation has finished as a function over time (i.e., the point (x, y) on the plot means that for y tasks, the symmetry precomputation time is $\leq x$ seconds).

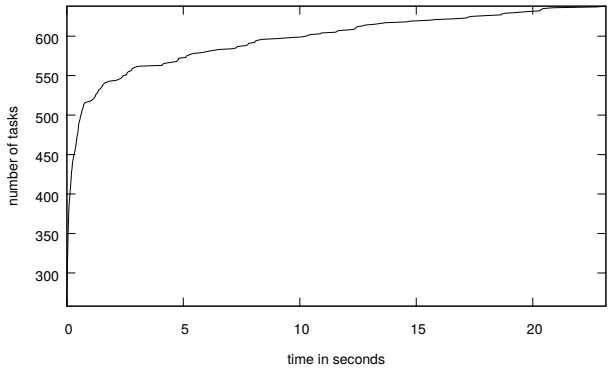

Figure 3: Number of planning tasks for which the symmetry precomputation finished as a function over time (in seconds)

We observe that the computation of symmetries can mostly be done rather efficiently: For most tasks, the computation finished within a couple of seconds (e.g., for $\geq 78\%$ of the considered tasks, the symmetry computation finished within one second, for $\geq 90\%$ of the tasks within ten seconds).

As a natural follow-up question, given that symmetries exist in many FOND planning domains, we investigate to which extent the size of the search spaces can be reduced when applying CLAO*, and whether the reduction pays off in lower runtime. To address these questions, we provide scatterplots for the number of generated nodes and the runtime (including the time for the symmetry precomputation) in Figure 4 and 5.

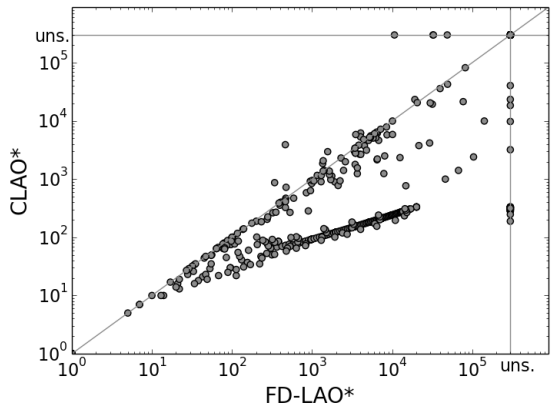

Figure 4: Number of node generations with FD-LAO* (x-axis) vs. CLAO*(y-axis).

A dot below the diagonal means that FD-LAO* generates more nodes (or needs more time, respectively) than CLAO*, and a dot above the diagonal means the opposite.

Figure 4 shows that the number of generated nodes with CLAO* is mostly as high as for FD-LAO* for most tasks. In addition, in several tasks CLAO* generates significantly fewer nodes, sometimes by several orders of magnitude. The "straight line" in the plot corresponds to tasks of the FOREST-NEW domain, where symmetry reduction particularly applies and reduces the state space.

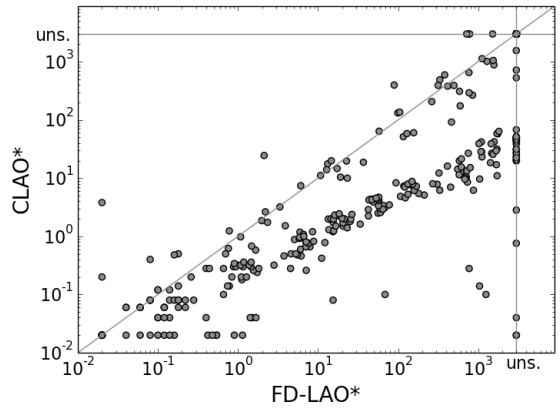

Figure 5: Runtime for FD-LAO* (x-axis) vs. CLAO*(y-axis).

Figure 5 shows that the reduced state space size in turn pays off in significantly reduced runtimes, again by orders of magnitude in several tasks. The runtimes include a slight overhead per node in order to compute symmetrical representative states. The plot shows in particular that this overhead is negligible compared to the smaller number of generated nodes.
Finally, we compare the coverage of FD-LAO* and CLAO* in Figure 6. The table particularly shows the total number of tasks per domain ($\#T$) as well as the number of tasks where symmetries have been found ($\#S$) in parenthesis ($\#S/\#T$).

	Coverage	
Domain	FD-LAO*	CLAO*
BLOCKSWORLD(15/30)	**23**	–2
BLOCKSWORLD-2(5/15)	**8**	±**0**
ELEVATORS(12/15)	**14**	±**0**
EX-BLOCKSWORLD(13/15)	8	**+1**
FAULTS-NEW(167/190)	167	**+21**
FAULTS(54/55)	52	**+3**
FIRST-RESPONDERS(97/100)	97	**+2**
FOREST(84/100)	**8**	±**0**
FOREST-NEW(88/100)	**7**	–1
TRIANGLE-TIREWORLD(0/40)	**7**	±**0**
Sum (535/660)	391	**+24**

Figure 6: Coverage result overview for FD-LAO* compared to CLAO*. In parenthesis: tasks where symmetries have been found / total number of tasks.

The fewer number of generated nodes of CLAO* also pays off in a significant coverage increase of 24 additionally solved tasks. Again, we remark that such a coverage increase is remarkable because the task's complexity usually increases exponentially in its size. In particular, CLAO* achieves a significant coverage increase in the FAULTS-NEW domain (21 additionally solved tasks). The domains FOREST-NEW and BLOCKSWORLD show that the per-node overhead of symmetry reduction can also lead to a slight coverage decrease.

Finally, we also compared the sizes of the policies i.e., the number of state-action pairs needed to store the strong cyclic plan found by FD-LAO* and CLAO*. The plot is depicted in Figure 7. We observe that the sizes are often similar, although canonical policies are sometimes significantly more compact.

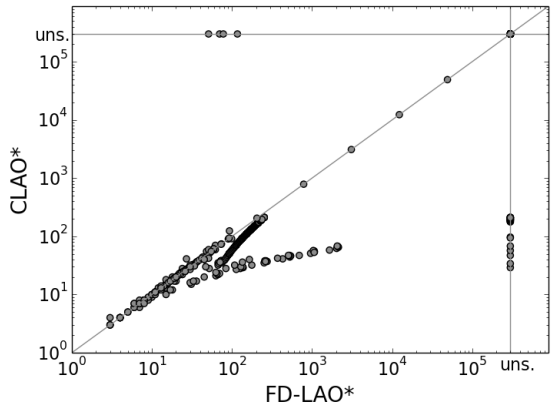

Figure 7: Sizes of policies found by FD-LAO* (x-axis) vs. CLAO*(y-axis).

Overall, our experiments show that symmetry reduction can be very beneficial for FOND planning: Symmetries do occur in various domains, they can mostly be computed efficiently, and the resulting smaller state spaces often allow CLAO* to solve more tasks than FD-LAO*.

As a side effect of our experiments, we have also observed that the base implementation FD-LAO* of LAO* in the Fast Downward planning system has already shown promising performance compared to myND. We have thus also presented a new and competitive LAO*-based FOND planner, making multiple Fast Downwards features available to the FOND planning community.

6 Conclusion

We have lifted symmetry reduction from classical planning with A* to FOND planning with LAO*. Our experimental evaluation has shown that the resulting algorithm CLAO* is beneficial compared to pure LAO* on a variety of common FOND planning domains. In the future, motivated by the encouraging results, it will be interesting to investigate the further potential of pruning methods for FOND planning with LAO*, with a particular focus on the question whether synergy effects of different pruning methods can be achieved.

Acknowledgments

This work was partially supported by the Swiss National Science Foundation (SNSF) as part of the project "Automated Reformulation and Pruning in Factored State Spaces (ARAP)".

References

[Bäckström and Nebel, 1995] Christer Bäckström and Bernhard Nebel. Complexity results for SAS^+ planning. *Computational Intelligence*, 11(4):625–655, 1995.

[Coles and Coles, 2010] Amanda Jane Coles and Andrew Coles. Completeness-preserving pruning for optimal planning. In *Proc. ECAI 2010*, pages 965–966, 2010.

[Domshlak *et al.*, 2012] Carmel Domshlak, Michael Katz, and Alexander Shleyfman. Enhanced symmetry breaking in cost-optimal planning as forward search. In *Proc. ICAPS 2012*, 2012.

[Domshlak *et al.*, 2013] Carmel Domshlak, Michael Katz, and Alexander Shleyfman. Symmetry breaking: Satisficing planning and landmark heuristics. In *Proc. ICAPS 2013*, pages 298–302, 2013.

[Domshlak *et al.*, 2015] Carmel Domshlak, Michael Katz, and Alexander Shleyfman. Symmetry breaking in deterministic planning as forward search: Orbit space search algorithm. Technical Report IS/IE-2015-03, Technion, Haifa, 2015.

[Fox and Long, 1999] Maria Fox and Derek Long. The detection and exploitation of symmetry in planning problems. In *Proc. IJCAI 1999*, pages 956–961, 1999.

[Fox and Long, 2002] Maria Fox and Derek Long. Extending the exploitation of symmetries in planning. In *Proc. AIPS 2002*, pages 83–91, 2002.

[Hansen and Zilberstein, 2001] Eric A. Hansen and Shlomo Zilberstein. LAO^*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1–2):35–62, 2001.

[Helmert, 2006] Malte Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[Hoffmann and Nebel, 2001] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[Holte and Burch, 2014] Robert C. Holte and Neil Burch. Automatic move pruning for single-agent search. *AI Communications*, 27(4):363–383, 2014.

[Holte *et al.*, 2015] Robert C. Holte, Yusra Alkhazraji, and Martin Wehrle. A generalization of sleep sets based on operator sequence redundancy. In *Proc. AAAI 2015*, pages 3291–3297, 2015.

[Mattmüller *et al.*, 2010] Robert Mattmüller, Manuela Ortlieb, Malte Helmert, and Pascal Bercher. Pattern database heuristics for fully observable nondeterministic planning. In *Proc. ICAPS 2010*, pages 105–112, 2010.

[Muise *et al.*, 2012] Christian J. Muise, Sheila A. McIlraith, and J. Christopher Beck. Improved non-deterministic planning by exploiting state relevance. In *Proc. ICAPS 2012*, pages 172–180, 2012.

[Muise *et al.*, 2014] Christian J. Muise, Sheila A. McIlraith, and Vaishak Belle. Non-deterministic planning with conditional effects. In *Proc. ICAPS 2014*, pages 370–374, 2014.

[Nissim *et al.*, 2012] Raz Nissim, Udi Apsel, and Ronen I. Brafman. Tunneling and decomposition-based state reduction for optimal planning. In *Proc. ECAI 2012*, pages 624–629, 2012.

[Pochter *et al.*, 2011] Nir Pochter, Aviv Zohar, and Jeffrey S. Rosenschein. Exploiting problem symmetries in state-based planners. In *Proc. AAAI 2011*, pages 1004–1009, 2011.

[Rintanen, 2003] Jussi Rintanen. Symmetry reduction for SAT representations of transition systems. In *Proc. ICAPS 2003*, pages 32–40, 2003.

[Shleyfman *et al.*, 2015] Alexander Shleyfman, Michael Katz, Malte Helmert, Silvan Sievers, and Martin Wehrle. Heuristics and symmetries in classical planning. In *Proc. AAAI 2015*, pages 3371–3377, 2015.

[Wehrle and Helmert, 2012] Martin Wehrle and Malte Helmert. About partial order reduction in planning and computer aided verification. In *Proc. ICAPS 2012*, 2012.

[Wehrle and Helmert, 2014] Martin Wehrle and Malte Helmert. Efficient stubborn sets: Generalized algorithms and selection strategies. In *Proc. ICAPS 2014*, pages 323–331, 2014.

[Wehrle *et al.*, 2013] Martin Wehrle, Malte Helmert, Yusra Alkhazraji, and Robert Mattmüller. The relative pruning power of strong stubborn sets and expansion core. In *Proc. ICAPS 2013*, pages 251–259, 2013.