Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling

Enhanced Symmetry Breaking in Cost-Optimal Planning as Forward Search

Carmel Domshlak
Technion
Haifa, Israel
dcarmel@ie.technion.ac.il

Michael Katz
Saarland University
Saarbrücken, Germany
katz@cs.uni-saarland.de

Alexander Shleyfman
Technion
Haifa, Israel
shleyfman.alexander@gmail.com

Abstract

In heuristic search planning, state-space symmetries are mostly ignored by both the search algorithm and the heuristic guidance. Recently, Pochter, Zohar, and Rosenschein (2011) introduced an effective framework for detecting and accounting for state symmetries within A^* cost-optimal planning. We extend this framework to allow for exploiting strictly larger symmetry classes, and thus pruning strictly larger parts of the search space. Our approach is based on exploiting information about the part of the transition system that is gradually being revealed by A^*. An extensive empirical evaluation shows that our approach allows for substantial reductions in search effort overall, and in particular, allows for more problems being solved.

Introduction

To date, A^* search with admissible heuristic functions is a prominent approach to cost-optimal planning. Numerous admissible heuristics for domain-independent planning have been proposed, varying from cheap to compute and not very informative to expensive to compute and very informative (Bonet and Geffner 2001; Haslum and Geffner 2000; Helmert, Haslum, and Hoffmann 2007; Katz and Domshlak 2010; Karpas and Domshlak 2009; Helmert and Domshlak 2009; Bonet and Helmert 2010). However, while further progress in developing informative heuristics is still very much desired, it is also well known that, on many problems, A^* expands an exponential number of nodes even if equipped with heuristics that are almost perfect in their estimates (Helmert and Röger 2008). One major reason for that is state symmetries in the transition systems of interest. A succinct description of the planning tasks in languages such as STRIPS and SAS$^+$almost unavoidably results in lots of different states in the search space to be symmetric to one another with respect to the task at hand. In turn, failing to detect and account for these symmetries results in A^* searching through many symmetric states, although searching through a state is equivalent to searches through all of its symmetric counterparts.

The idea of identifying and pruning symmetries while reasoning about automorphisms of the search spaces has been exploited for quite a while already in model checking (Emerson and Sistla 1996), constraint satisfaction (Puget 1993), and planning (Rintanen 2003; Fox and Long 1999; 2002). However, until the recent work by Pochter et al. (2011), no empirical successes in this direction have been reported in the scope of cost-optimal planning as heuristic forward search. The success of the framework proposed by Pochter et al. is especially valuable because, to date, heuristic forward search with A^* constitutes the most effective approach to cost-optimal planning.

In this work, we build upon the framework of Pochter et al. (2011) and extend it to allow for exploiting strictly larger sets of automorphisms, and thus pruning strictly larger parts of the search space. Our approach is based on exploiting information about the part of the transition system that is gradually being revealed by the A^* algorithm. This information allows us to eliminate the requirement of Pochter et al. from the automorphisms to stabilize the initial state, a requirement that turns out to be quite constraining in terms of state-space pruning. We introduce a respective extension of the A^* algorithm that preserves its core properties of completeness and optimality. Similarly to the work of Pochter et al. , our approach works at the level of the search algorithm, and is completely independent of the heuristic in use. Our empirical evaluation shows that our approach to A^* symmetry breaking favorably competes with the previous work of Pochter et al. (2011), increasing the number of problems solved, and significantly reducing the search effort required to solve planning tasks.

Preliminaries

We consider classical planning tasks $\Pi = \langle V, A, s_0, G\rangle$ captured by the well-known SAS$^+$ formalism (Bäckström and Nebel 1995). In such a task, V is a set of finite-domain *state variables*; each complete assignment to V is called a *state*, and $S = \prod_{v \in V} dom(v)$ is the *state space* of Π. s_0 is an *initial state*. The *goal* G is a partial assignment to V; a state s is a *goal state*, denoted by $s \in S_*$, iff $G \subseteq s$. A is a finite set of *actions*, each given by a pair $\langle \mathsf{pre}, \mathsf{eff}\rangle$ of partial assignments to V, called *preconditions* and *effects*. Applying action a in state s results in a state denoted by $s[\![a]\!]$.

Our focus here is on cost-optimal planning, and we assume familiarity with the standard A^* search algorithm. By $\mathcal{T} = \langle S, E\rangle$ we refer to the state transition (di)graph induced by Π, with parallel edges induced by state-wise equivalent actions being represented by a single edge in E. Auxiliary notation: $i \in [k]$ is used for $i \in \{1, 2, \ldots, k\}, k \in \mathbb{N}$.

An **automorphism** of a transition graph $\mathcal{T} = \langle S, E \rangle$ is a permutation σ of the vertices S such that $(s, s') \in E$ iff $(\sigma(s), \sigma(s')) \in E$. The composition of two automorphisms is another automorphism, and the complete, under the composition operator, set of automorphisms forms the **automorphism group** $Aut(\mathcal{T})$ of the graph. By $\Gamma \leq \Gamma'$ we denote that Γ is a *subgroup* of Γ'. By σ_{id} we denote the trivial automorphism that is the identity morphism. Each subgroup of automorphisms $\Gamma \leq Aut(\mathcal{T})$ induces an **equivalence** relation $\sim_\Gamma$ on states S: $s \sim_\Gamma s'$ iff there exists σ in Γ such that $\sigma(s) = s'$. For a pair of equivalence relations $\sim_1$ and $\sim_2$ on states S, if $s \sim_1 s'$ implies $s \sim_2 s'$, we say that $\sim_2$ is *coarser* than $\sim_1$ ($\sim_1$ is *finer* than $\sim_2$), and denote that by $\sim_1 \leq \sim_2$. Among all the subgroups of automorphisms, $Aut(\mathcal{T})$ itself induces the coarsest equivalence relation $\sim^*$ that separates all the non-symmetrical states of $\mathcal{T}$.

Let $S_1, \ldots, S_k$ be some subsets of S and $\Gamma \leq Aut(\mathcal{T})$ be a subgroup of automorphisms. Let $\Gamma_{S_1,\ldots,S_k} = \{\sigma \in \Gamma \mid \forall i \in [k], \forall s \in S_i : \sigma(s) \in S_i\}$ be the set of elements of Γ that map states in each S_i only to states in S_i. Then $\Gamma_{S_1,\ldots,S_k}$ is a subgroup of Γ called the **stabilizer** of state subsets $S_1, \ldots, S_k$ with respect to $\Gamma \leq Aut(\mathcal{T})$. Finally, a set of automorphisms Σ is said to **generate** a group Γ if Γ is the fixpoint of iterative composition of the elements of Σ. Finding such a generating set of $Aut(\mathcal{T})$ (for an explicitly given graph $\mathcal{T}$) is not known to be polynomial-time, but backtracking search techniques are surprisingly effective in finding generating sets for substantial subgroups $\Gamma \leq Aut(\mathcal{T})$ (Junttila and Kaski 2007).

Previous Work: State Pruning with $\Gamma_{\{s_0\},S_*}$

In our work we build upon the approach by Pochter et al. (2011) for exploiting state space symmetries in cost-optimal planning using A^*. Here we describe the key aspects of this approach, referring the reader for further details to the original publication.

The basic principle underlying the work of Pochter et al. is similar to the idea of canonicalization by Emerson and Sistla (1996), formalized in the context of the A^* algorithm: For any group $\Gamma \leq \Gamma_{\{s_0\},S_*} \leq Aut(\mathcal{T})$, and any equivalence relation $\sim \leq \sim_\Gamma$, a state s can be pruned without forfeiting the optimality and completeness of A^*, as long as some other state in its equivalence class defined by $\sim$ is expanded. The modification of A^* required for exploiting this observation is as follows.

1. Offline: Find a subset Σ of generators for the group $\Gamma_{\{s_0\},S_*}$. Let $\Gamma \leq \Gamma_{\{s_0\},S_*}$ be the group generated by Σ. Using Σ, find an equivalence relation $\sim \leq \sim_\Gamma$.
2. Whenever search generates a state s such that $s \sim s'$ for some previously generated state s', treat s as if it was s'.

This is, of course, more of a template for A^* modification, and one has to specify efficient means for (i) determining the equivalence relation $\sim$ using a generating set Σ, and, what is even more challenging, (ii) finding the actual generating set Σ. The concrete proposals of Pochter et al. for accomplishing these two tasks are the key components of their contribution, and we discuss their essence in what comes next.

As the state transition graph $\mathcal{T}$ of a SAS$^+$ planning task Π is not given explicitly, automorphisms of $\mathcal{T}$ must be inferred from the description of Π. The specific method that Pochter et al. proposed for deducing automorphisms of $\mathcal{T}$ exploits automorphisms of a certain graphical structure induced by the task description. This node-colored undirected graph, called problem description graph (PDG), is in particular convenient for discovery of automorphisms of $\mathcal{T}$ that are stabilized with respect to an arbitrary set of state sets $S_1, \ldots, S_m \subseteq S$ where each $S_i = \{s \in S \mid s[V_i] = p_i\}$ is characterized by a single partial assignment to the state variables V. [1] This property of PDGs in particular allows for searching for generators of $\Gamma_{\{s_0\},S_*}$ using off-the-shelf tools for discovery of automorphisms in explicit, node-colored graphs such as BLISS (Junttila and Kaski 2007).

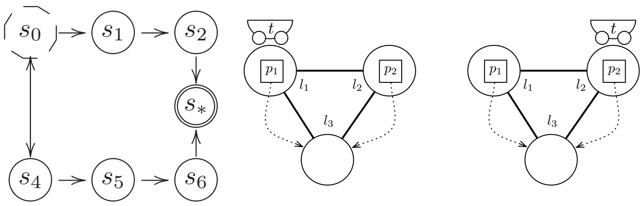
Having discovered this way a generating set of a group $\Gamma \leq \Gamma_{\{s_0\},S_*}$, the next step is to determine an equivalence relation $\sim$. Each equivalence class of $\sim$ is *implicitly* represented by one of its states $s^\dagger$, called the canonical state. While finding the coarsest relation $\sim_\Gamma$ is NP-hard (Luks 1993), nothing in the above modification of A^* requires this relation to be the coarsest one. The specific approach suggested by Pochter et al. constitutes a *procedural* mapping $C : S \to S$ from states to states in their equivalence class. This mapping C is implemented via a heuristic local search in a space with states being our planning task states S, actions corresponding to the generators Σ, and state evaluation being based on a lexicographic ordering of S. Each local minimum in this space defines an equivalence class by "defining itself" to be the canonical state $s^\dagger$ of that class. In other words, two states s and s' are equivalent if their canonical states are the same, that is $s \sim s'$ iff $C(s) = C(s')$.

The empirical results obtained by Pochter et al. (2011) by modifying the A^* algorithm implementation of the Fast Downward planner (Helmert 2006) were truly impressive. First, the modified A^* substantially outperformed the basic A^* in terms of the solved IPC benchmarks, and this across numerous heuristic functions, from the trivial blind heuristic to state-of-the-art abstraction and landmark heuristics. At least as importantly, (i) the improvement was achieved in numerous domains, and not only on the extremely symmetric GRIPPER domain, and (ii) the improvement was robust in the sense that, on no domain the time overhead of exploiting $\Gamma_{\{s_0\},S_*}$ symmetries resulted in solving less tasks than with the basic A^*. In what follows, we show that the envelope of exploiting graph automorphisms in cost-optimal planning with A^* can be pushed even further.

A^* Symmetry Breaking with Γ_{S_*}

We now proceed with describing our approach, and we begin with discussing a couple of motivating examples. Consider the state transition graph depicted in Figure 1a. For this

[1] Our definition of PDG differs from the original one by Pochter et al. (2011) in that action nodes are colored, to distinguish between actions of different cost. This modification, however, is truly minor.



(a) (b)

Figure 1: Illustrations for the examples on Γ_{S_*} vs. $\Gamma_{\{s_0\},S_*}$.

graph, the group $\Gamma_{\{s_0\},S_*}$ of stabilizers with respect to both the initial state and the goal consists of only the trivial automorphism σ_{id}, and thus it will not be useful for pruning. In particular, A^* with blind heuristic will examine all the states $s_0, \ldots, s_6$ before reaching the goal state s_*. Note, however, that there exists $\sigma \in \Gamma_{S_*}$ such that $s_1 = \sigma(s_5)$, and thus, in particular, $h^*(s_1) = h^*(s_5)$. Hence, if A^* happens to generate state s_5 after generating state s_1, then s_5 can be safely pruned without violating optimality of the search. The picture, of course, is not symmetric, and if s_1 is discovered after s_5, then s_1 cannot be pruned as it may lie on the only optimal plan for the task (which is actually the case in this schematic example). Later we show that something can be done about s_1 even in such situations.

To further illustrate the direction suggested by this schematic example, consider a simple LOGISTICS example with three, fully connected locations l_1, l_2, l_3, two packages, and a single truck. The packages p_1 and p_2 are initially at l_1 and l_2, respectively, and the goal is to bring both packages to l_3. Now, if the truck is initially at l_3, then the two states depicted in Figure 1b are symmetric with respect to $\Gamma_{\{s_0\},S_*}$, and thus only one of them will be expanded by A^* modified as in the previous section. However, if the truck is initially at l_1, that is, s_0 is the state on the left of Figure 1b, then the two states are no longer symmetric with respect to $\Gamma_{\{s_0\},S_*}$; in fact, $\Gamma_{\{s_0\},S_*}$ now consists of only the trivial σ_{id}. It is, however, very unreasonable to expand the state on the right of Figure 1b as it is symmetric with respect to Γ_{S_*} to the *initial state* of the task.

A simple property of plans and state automorphisms that generalizes the intuition provided by the examples above is as follows: Let Π be a planning task, Γ be a subgroup of Γ_{S_*}, and $(s_0, s_1, \ldots, s_k)$, $(s_0, s'_1, \ldots, s'_l)$ be a pair of plans for Π, that is, $\{s_k, s'_l\} \subseteq S_*$. If, for some $i \in [k]$ and $i < j \in [l]$, $s_i = \sigma(s'_j)$ for some $\sigma \in \Gamma$, then $(s_0, \ldots, s_{i-1}, \sigma(s'_j), \ldots, \sigma(s'_l))$ is also a plan for Π, shorter than the plan $(s_0, s'_1, \ldots, s'_l)$.

While not very prescriptive in itself, this property leads to the following observation about the prospects of exploiting state automorphisms within A^*: In above terms, if A^* generates s_i before generating s'_j, then s'_j can safely be pruned. Moreover, if A^* generates s_i *after* s'_j, then we can still "prune" s_i and continue working with s'_j and its successors, as long as we memorize that s'_j is no longer represents itself, but its Γ_{S_*}-symmetric counterpart s_i.

The modification of A^* required for exploiting this observation is described in Figure 2; notation $g^{\text{tmp}}(s)$, $\mathsf{parent}(s)$, and $\mathsf{act}(s)$ capture standard search-node information associated with state s, respectively, distance-so-far, parent state,

1. Offline: Find a subset Σ of generators for the group Γ_{S_*}. Let $\Gamma \leq \Gamma_{S_*}$ be the group generated by Σ. Using Σ, find an equivalence relation $\sim \leq \sim_\Gamma$.
2. Whenever search generates a state s such that $s \sim s'$ for some previously generated state s', if $g^{\text{tmp}}(s) < g^{\text{tmp}}(s')$, then set $g^{\text{tmp}}(s') := g^{\text{tmp}}(s)$, $\mathsf{parent}(s') := \mathsf{parent}(s)$, and $\mathsf{act}(s') := \mathsf{act}(s)$, and then reopen s'. Otherwise, if $g^{\text{tmp}}(s) \geq g^{\text{tmp}}(s')$, prune s as if it was never generated.
3. If a goal state s_* is reached, (i) extract a sequence of pairs of state and action $\pi = \langle(\varepsilon, s_0), (a_1, s_1), \ldots, (a_m, s_m)\rangle$, where $s_m = s_*$, by the standard backchaining from s_* along the parent relation, setting actions by the act relation, and (ii) generate a valid plan from π using $\mathsf{trace\text{-}forward}(\pi)$.

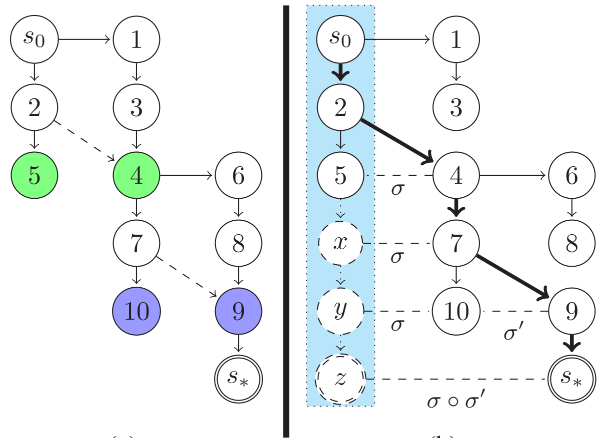
$\mathsf{trace\text{-}forward}(\pi)$:
let $\pi = \langle(\varepsilon, s_0), (a_1, s_1), \ldots, (a_m, s_m)\rangle$, and, for $i \in [m]$,
let $\sigma_i \in \Gamma_{S_*}$ be such that $\sigma_i(s_i) = s_{i-1}[\![a_i]\!]$
$\sigma := \sigma_{\text{id}}$, $\rho := \langle\varepsilon\rangle$
for $i := 1$ **to** m **do**
  $s := \sigma(s_{i-1})$, $\sigma := \sigma_i \circ \sigma$, $s' := \sigma(s_i)$
  append to ρ a cheapest action a such that $s[\![a]\!] = s'$
return ρ

Figure 2: A^* modification for symmetry breaking with Γ_{S_*}.

and action using which s is obtained from its parent. Actually, the core search mechanism of A^* remains unchanged, and at high level, step 2 can be summarized as: Whenever search generates a state s such that $s = \sigma(s')$ for some previously generated state s' and some $\sigma \in \Gamma$, treat s as if it was s'. However, when the current path to s is shorter than the current path to s', the parent s_p of s "adopts" s' as a pseudo-child while we memorize the action a such that $s = s_p[\![a]\!]$. The major difference of the algorithm here from the plain A^* is in the plan extraction routine. Plan extraction in A^* is done simply by backchaining from the discovered goal state to the initial state along the parent connections. In contrast, with the modified algorithm, some of these connections may correspond to adoptions, that is, the chain π of action/state pairs provided by the standard backchaining may correspond neither to a plan for Π, nor even to an action sequence applicable in s_0.

While this is indeed so, π can still be efficiently converted into a plan for Π, using the trace-forward procedure in Figure 2. The only not self-explanatory step there is determining mappings σ_i for $i \in [m]$: If $s_{i-1}[\![a_i]\!] = s_i$, that is, s_{i-1} is the true parent of s_i, then $\sigma_i = \sigma_{\text{id}}$. Otherwise, we still have $s_i \sim s_{i-1}[\![a_i]\!]$, and let $\sigma_{[i,1]}(s_i) = \sigma_{[i,2]}(s_{i-1}[\![a_i]\!]) = s^\dagger$, where $\sigma_{[i,1]}$ and $\sigma_{[i,2]}$ are determined using the local search procedure C. Then, $\sigma_i = \sigma_{[i,2]}^{-1} \circ \sigma_{[i,1]}$.

The example depicted in Figure 3 illustrates this "plan reconstruction" procedure. Figure 3a depicts the part of the search space generated before the search reached the goal state s_*; the states are numbered in the order of their generation and solid arcs capture the successor relation be-



(a) (b)

Figure 3: Illustration of search result and plan extraction.

tween the states. Assume now that $5 \sim 4$ and $10 \sim 9$. Given that, when 5 is generated, $\mathsf{parent}(4)$ is switched from the true parent 3 to the "adopting parent" 2, and similarly, $\mathsf{parent}(9)$ is switched from 8 to 7. These adoption relations are depicted in Figure 3a by dashed arcs. Figure 3b depicts the backchained pseudo-plan π (thick arcs), as well as the valid plan ρ, reconstructed from π in $\mathsf{trace\text{-}forward}$ (filled rectangle). Only states s_0 and 2 are shared by π and ρ; states 5 and x are reconstructed from states 4 and 7, respectively, via state automorphism $\sigma \in \Gamma_{S_*}$ for which $\sigma(4) = 5 = \mathsf{parent}(4)[\![\mathsf{act}(4)]\!] = 2[\![\mathsf{act}(4)]\!]$, and states y and (goal state) z are reconstructed from 9 and s_*, respectively, via *composition* of σ with $\sigma' \in \Gamma_{S_*}$ for which $\sigma'(9) = 10 = \mathsf{parent}(9)[\![\mathsf{act}(9)]\!] = 7[\![\mathsf{act}(9)]\!]$.

Performance Evaluation

We implemented our approach on top of the Fast Downward planner (Helmert 2006), and evaluated it on all the applicable benchmarks from IPC 1–6. The comparison was made both to the approach of Pochter et al. we build upon, as well as to the plain A^* with no symmetry breaking. All of the experiments were run on Intel E8200 with the standard time limit of 30 minutes and memory limit of 2 GB.

Figure 4 depicts the results for our approach (Γ_{S_*}), previous approach ($\Gamma_{\{s_0\},S_*}$), and plain A^* in the context of blind and LM-cut (Helmert and Domshlak 2009) heuristics. The experiments with the blind heuristic aim at distilling the potential of the symmetry breaking techniques, while the experiments with the state-of-the-art LM-cut aim at realizing the marginal contribution of symmetry breaking on top of a relatively high-quality search guidance. The results are presented in terms of both the number of problems solved ("s") and in terms of the node expansions until the solution ("E"). The metric score of the number of node expansions for configuration c on some problem is E^*/E_c, where E_c is the number of nodes expanded under configuration c, and E^* is the best (minimal) number of node expansions by any configuration on that problem. Thus the best value for each problem is assigned a metric score of 1, and expanding twice as many nodes would lead to a score of 0.5; not solving the problem results in a score of 0. We report the total score for

Domain	SA	Γ_{S_*}		$\Gamma_{\{s_0\},S_*}$		No symm	
		s	E	s	E	s	E
airport	19	**19**	**18.98**	**19**	17.99	18	15.85
blocks	18	**18**	**18.00**	**18**	**18.00**	**18**	**18.00**
depot	5	**5**	**5.00**	4	2.30	4	1.91
driverlog	7	**7**	**7.00**	**7**	5.03	**7**	3.02
elevators	11	**11**	**11.00**	**11**	9.85	**11**	9.85
freecell	14	**14**	**14.00**	**14**	**14.00**	**14**	13.53
grid	1	**1**	**1.00**	**1**	0.88	**1**	0.88
gripper	20	**20**	**20.00**	**20**	**20.00**	7	0.13
logistics00	10	**10**	**10.00**	**10**	7.72	**10**	5.29
logistics98	2	**2**	**2.00**	**2**	0.71	**2**	0.71
miconic	51	**51**	**51.00**	**51**	44.57	50	27.17
mprime	20	19	16.23	**20**	**19.86**	19	14.82
mystery	19	**19**	17.20	18	**17.22**	18	14.54
openstacks08	23	**23**	**23.00**	**23**	22.63	18	5.68
openstacks06	7	**7**	**7.00**	**7**	**7.00**	**7**	3.44
parcprinter	10	**10**	**10.00**	**10**	**10.00**	**10**	**10.00**
pathways	4	**4**	**4.00**	**4**	**4.00**	**4**	2.94
pegsol	27	**27**	25.31	**27**	**26.12**	**27**	22.68
pipesworld-nt	15	**15**	**15.00**	14	8.25	13	4.47
pipesworld-t	17	**17**	**17.00**	14	5.40	10	2.86
psr-small	49	**49**	**49.00**	**49**	46.17	**49**	36.69
rovers	5	**5**	**5.00**	**5**	**5.00**	**5**	4.63
satellite	6	**6**	**6.00**	5	3.02	4	1.16
scanalyzer	13	12	**12.00**	**13**	11.68	**13**	6.53
sokoban	22	**22**	**21.89**	18	10.51	18	9.53
tpp	6	**6**	**6.00**	**6**	5.61	5	2.41
transport	11	**11**	**10.93**	**11**	8.98	**11**	7.88
trucks	5	**5**	**5.00**	**5**	4.58	**5**	4.14
woodworking	7	**7**	**7.00**	**7**	**7.00**	**7**	6.53
zenotravel	8	**8**	**8.00**	**8**	6.21	7	4.94
	432	**430**	**423.54**	421	370.27	392	262.22

(a)

Domain	SA	Γ_{S_*}		$\Gamma_{\{s_0\},S_*}$		No symm	
		s	E	s	E	s	E
airport	28	**28**	**28.00**	**28**	27.92	**28**	27.92
blocks	28	**28**	**28.00**	**28**	**28.00**	**28**	**28.00**
depot	8	**8**	**8.00**	7	5.24	7	4.37
driverlog	13	**13**	**13.00**	**13**	12.53	**13**	10.98
elevators	22	**22**	**22.00**	**22**	20.64	**22**	20.06
freecell	15	**15**	**15.00**	**15**	**15.00**	**15**	14.55
grid	2	**2**	**2.00**	**2**	1.96	**2**	1.96
gripper	20	**20**	19.27	**20**	**20.00**	7	0.21
logistics00	20	**20**	19.45	**20**	**19.98**	**20**	16.57
logistics98	6	**6**	**5.98**	**6**	5.63	**6**	3.96
miconic	141	**141**	**141.00**	**141**	140.86	**141**	137.46
mprime	23	**23**	21.15	**23**	**23.00**	**23**	20.48
mystery	21	**21**	**20.17**	**21**	19.97	**21**	18.58
openstacks08	23	**23**	**23.00**	**23**	22.64	18	5.69
openstacks06	7	**7**	**7.00**	**7**	**7.00**	**7**	3.45
parcprinter	18	**18**	**18.00**	**18**	**18.00**	**18**	**18.00**
pathways	5	**5**	**5.00**	**5**	**5.00**	**5**	4.83
pegsol	28	**28**	26.20	**28**	**27.79**	27	23.31
pipesworld-nt	20	**20**	**19.60**	18	15.43	17	8.44
pipesworld-t	16	**16**	**16.00**	13	7.15	11	3.70
psr-small	50	**50**	**50.00**	49	47.52	49	39.07
rovers	7	**7**	**7.00**	**7**	**7.00**	**7**	6.95
satellite	12	**12**	**11.99**	10	9.04	7	5.02
scanalyzer	16	**16**	**15.92**	**16**	15.83	15	11.86
sokoban	29	28	**27.02**	**29**	19.93	28	17.10
tpp	7	**7**	**7.00**	**7**	6.97	6	4.37
transport	11	**11**	**11.00**	**11**	10.43	**11**	9.86
trucks	10	**10**	**10.00**	**10**	9.93	**10**	9.56
woodworking	17	**17**	**17.00**	**17**	16.76	16	13.66
zenotravel	13	**13**	12.63	**13**	**12.93**	**13**	11.77
	636	**635**	**627.36**	627	600.09	598	501.72

(b)

Figure 4: Number of solved tasks and efficiency in terms of expanded nodes with (a) blind and (b) LM-cut heuristics.

each domain, as well as the total score overall, and this over all problems solved by some configuration ("SA").

The results clearly testify for the increased effectiveness of the new symmetry breaking method in terms of the number of problems solved, but probably even more so, in terms of the search effort required to solve individual problems. Overall, however, we believe that further progress can be achieved in symmetry breaking for state-space search. The message we hope our results communicate is that the key for that progress might be in exploiting the information that either can be or is already collected by the search algorithms.

References

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS$^+$ planning. *Comp. Intell.* 11(4):625–655.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *AIJ* 129(1–2):5–33.

Bonet, B., and Helmert, M. 2010. Strengthening landmark heuristics via hitting sets. In *ECAI*, 329–334.

Emerson, E. A., and Sistla, A. P. 1996. Symmetry and model checking. *Formal Methods in System Design* 9(1–2):105–131.

Fox, M., and Long, D. 1999. The detection and exploitation of symmetry in planning problems. In *IJCAI*, 956–961.

Fox, M., and Long, D. 2002. Extending the exploitation of symmetries in planning. In *AIPS*, 83–91.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *ICAPS*, 140–149.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *ICAPS*, 162–169.

Helmert, M., and Röger, G. 2008. How good is almost perfect? In *AAAI*, 944–949.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In *ICAPS*, 200–207.

Helmert, M. 2006. The Fast Downward planning system. *JAIR* 26:191–246.

Junttila, T., and Kaski, P. 2007. Engineering an efficient canonical labeling tool for large and sparse graphs. In *ALENEX*, 135–149.

Karpas, E., and Domshlak, C. 2009. Cost-optimal planning with landmarks. In *IJCAI*, 1728–1733.

Katz, M., and Domshlak, C. 2010. Implicit abstraction heuristics. *JAIR* 39:51–126.

Luks, E. M. 1993. Permutation groups and polynomial-time computation. In *Groups and Computation, DIMACS Series in Disc. Math. and Th. Comp. Sci.*, volume 11. 139–175.

Pochter, N.; Zohar, A.; and Rosenschein, J. S. 2011. Exploiting problem symmetries in state-based planners. In *AAAI*, 1004–1009.

Puget, J.-F. 1993. On the satisfiability of symmetrical constrained satisfaction problems. In *ISMIS*, 350–361.

Rintanen, J. 2003. Symmetry reduction for SAT representations of transition systems. In *ICAPS*, 32–41.