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 et al. (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 $S = \prod_{v \in V} \text{dom}(v)$ is the state space of $\Pi$. $s_0$ 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 $\langle \text{pre}, \text{eff} \rangle$ of partial assignments to $V$, called preconditions and effects. Applying action $a$ in state $s$ results in a state denoted by $s \uplus a$.

Our focus here is on cost-optimal planning, and we assume familiarity with the standard $A^*$ search algorithm. By $T = \langle S, E \rangle$ we refer to the state transition (di)graph induced...
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

\footnote{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.}
1. Offline: Find a subset $\Sigma$ of generators for the group $\Gamma_{S_*}$. Let $\Gamma \leq \Gamma_{S_*}$ be the group generated by $\Sigma$. Using $\Sigma$, find an equivalence relation $\sim$.

2. Whenever search generates a state $s$ such that $s \sim s'$ for some previously generated state $s'$, if $g^\text{imp}(s) < g^\text{imp}(s')$, then set $g^\text{imp}(s') := g^\text{imp}(s)$, parent$(s') := \text{parent}(s)$, and act$(s') := \text{act}(s)$, and then reopen $s'$. Otherwise, if $g^\text{imp}(s) \geq g^\text{imp}(s')$, prune $s$ as if it was never generated.

3. If a goal state $s_*$ is reached, (i) extract a sequence of pairs of states and action $\pi = (\langle \varepsilon, s_0 \rangle, (a_1, s_1), \ldots, (a_m, s_m)\rangle$, 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 $\pi$ using trace-forward($\pi$).

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

return $\rho$

Figure 2: $A^*$ modification for symmetry breaking with $\Gamma_{S_*}$, 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 $\pi$ of action/state pairs provided by the standard backchaining may correspond neither to a plan for $\Pi$, nor even to an action sequence applicable in $s_0$.

While this is indeed so, $\pi$ can still be efficiently converted into a plan for $\Pi$, using the trace-forward procedure in Figure 2. The only not self-explanatory step there is determining mappings $\sigma_i$ 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 $\sigma_i = \sigma_{\delta}$. Otherwise, we still have $s_i \sim s_i-1[[a_i]]$, and let $\sigma_{[i,1]}(s_i) = \sigma_{[i,2]}(s_i-1[[a_i]]) = s_i$, where $\sigma_{[i,1]}$ and $\sigma_{[i,2]}$ are determined using the local search procedure $C$. Then, $\sigma_i = \sigma_{[i,2]} \circ \sigma_{[i,1]}$.

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-

Figure 1: Illustrations for the examples on $\Gamma_{S_*}$ 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 $\sigma_{\delta}$, 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 $s_1 = \sigma(s_0)$, and thus, in particular, $h^*(s_1) = h^*(s_0)$. 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 $\Gamma_{\{s_0\},S_*}$; in fact, $\Gamma_{\{s_0\},S_*}$ now consists of only the trivial $\sigma_{\delta}$. It is, however, very unreasonable to expand the state on the right of Figure 1b as it is symmetric with respect to $\Gamma_{S_*}$ 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 $\Pi$ be a planning task, $\Gamma$ be a subgroup of $\Gamma_{S_*}$, and $(s_0, s_1, \ldots, s_k), (s_0, s'_1, \ldots, s'_l)$ be a pair of plans for $\Pi$, that is, $(s_k, s'_l) \subseteq S_*$. If, for some $i \in [k]$ and $j < j' \in [l]$, $s_i = \sigma(s'_j)$ for some $\sigma \in \Gamma$, then $(s_0, \ldots, s_i-1, \sigma(s'_j), \ldots, \sigma(s'_l))$ is also a plan for $\Pi$, 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 $s_i$ and its successors, as long as we memorize that $s'_j$ is no longer represents itself, but its $\Gamma_{S_*}$-symmetric counterpart $s_i$.

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

We implemented our approach on top of the Fast Downward planner (Helmer 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 for \( \Gamma_{\Sigma_{\pi}} \), previous approach \( \Gamma_{\{\text{no}\}} \), \( \Sigma_{\pi} \), and plain \( A^* \) in the context of blind and LM-cut (Helmer 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 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.

Figure 3: Illustration of search result and plan extraction.
References


