# Symbolic Merge-and-Shrink for Cost-Optimal Planning

**Álvaro Torralba** and **Carlos Linares López** and **Daniel Borrajo**
Planning and Learning Group
Universidad Carlos III de Madrid
Leganés (Madrid) - Spain
{*alvaro.torralba, carlos.linares, daniel.borrajo*}*@uc3m.es*

## Abstract

Symbolic PDBs and Merge-and-Shrink (M&S) are two approaches to derive admissible heuristics for optimal planning. We present a combination of these techniques, Symbolic Merge-and-Shrink (SM&S), which uses M&S abstractions as a relaxation criterion for a symbolic backward search. Empirical evaluation shows that SM&S has the strengths of both techniques deriving heuristics at least as good as the best of them for most domains.

## 1 Introduction

One of the most successful approaches to cost-optimal planning is the use of abstraction heuristics as admissible estimations in an A$^*$ search. Abstraction heuristics use the optimal cost in a simplified abstract state space as an admissible estimation of the cost of the original problem. Merge-and-Shrink (M&S) is a flexible approach to derive abstractions [Helmert *et al.*, 2007]. It has been shown to be more general than other approaches such as Pattern Databases (PDBs) [Culberson and Schaeffer, 1998; Edelkamp, 2001] and it is able to derive the perfect heuristic for some domains [Nissim *et al.*, 2011a]. Moreover, it has shown impressive practical results, being the runner-up in the optimal sequential track of the 2011 International Planning Competition[1] (IPC) and a component of the portfolio winning the competition [Helmert *et al.*, 2011].

On the other hand, symbolic search performs state space exploration by representing sets of states in the form of Binary Decision Diagrams (BDDs) [Bryant, 1986]. The GAMER planner uses symbolic variants of bidirectional blind search and A$^*$ [Kissmann and Edelkamp, 2011]. GAMER won the optimal track of IPC-2008 and, even though it was not among the best performing planners in the IPC-2011, it showed potential being the best planner in some particular domains. GAMER generates heuristics in the form of Symbolic Pattern Databases by performing a symbolic backward Dijkstra's search on the abstracted state space. When the state space is not abstracted, a perimeter around the goal is explored. Thus, symbolic backward search and M&S may both generate perfect heuristics in a symbolic representation. Indeed, they have been shown to have equivalent representational power under the assumption of a linear merge strategy [Edelkamp *et al.*, 2012]. The difference lies on how they generate the heuristic. While M&S considers an abstract state space and iteratively adds more variables, symbolic search performs regression on the original state space.

We present Symbolic M&S (SM&S), a new algorithm to derive admissible heuristics for optimal planning. SM&S uses M&S abstractions as a relaxation criterion for a symbolic backward search. Thus, SM&S generates more accurate heuristics by combining the strengths of both techniques. The next section gives a formal definition of symbolic search and M&S. Then, SM&S is presented along with its theoretical properties. Finally, SM&S has been empirically evaluated and compared to previous state-of-the-art M&S strategies.

## 2 Background

Following the notation of [Katz *et al.*, 2012] a *planning task* is a tuple $\Pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ where $\mathcal{V}$ is a set of finite-domain *variables* $v$ with domains $D_v$, $\mathcal{O}$ is a set of *operators* $o = (pre, eff, c \in \mathbb{N}_0)$ composed of its preconditions, effects and cost, $s_0$ is an assignment of values to variables in $\mathcal{V}$ representing the *initial state* and $s_\star$ is a partial-assignment of values to a subset of variables denoted as the *goal*. A planning task defines a *state space* as a labeled transition system $\Theta = (S, L, T, s_0, S_\star)$, where $S$ is the set of all states. $L$ is a set of transition labels corresponding to the operators of the planning task. $T$ is the set of transitions, where each transition is a tuple $(s, l, s')$ with $s, s' \in S$ and $l \in L$ and $(s, l(o), s') \in T$ if $o$ is applicable in $s$, giving $s'$ as result. Finally, $s_0$ is the initial state and $S_\star$ is the set of states satisfying $s_\star$. A *plan* is an operator sequence describing a path from $s_0$ to any state in $S_\star$. A plan is *optimal* iff the summed costs of its operators is minimum. A *heuristic* is a function $h$: $s \to \mathbb{N}_0$ which estimates the cost to reach a state of $S_\star$ from a state $s$. The heuristic is *perfect* if it equals the optimal cost $h^*(s)$ for all $s \in S$. It is *admissible* if it never overestimates the optimal cost, $\forall s\ h(s) \leq h^*(s)$. It is *consistent* if for every $(s, l, s') \in T$, $h(s) \leq h(s') + c(l)$.

An *abstraction* is a function $\alpha$: $S \to S^\alpha$ mapping states in $S$ to abstract states in $S^\alpha$. The *abstract state space* of $\alpha$ is defined as a tuple $\Theta^\alpha = (S^\alpha, L, T^\alpha, s_0^\alpha, S_\star^\alpha)$ where $S^\alpha$ is the set of abstract states, $L$ is a set of labels, $T^\alpha = \{(\alpha(s), l, \alpha(s')) \mid (s, l, s') \in T\}$, $s_0^\alpha = \alpha(s_0)$

[1] `www.ipc.icaps-conference.org`

and $S^{\alpha}_{\star} = \{s^{\alpha} | \exists s \in S_{\star}, s^{\alpha} = \alpha(s)\}$. An *abstraction heuristic* $h_{\alpha}$ uses the cost of the cheapest path from $\alpha(s)$ to $S^{\alpha}_{\star}$. The set of *relevant variables* for $\alpha$ is denoted as $\mathcal{V}_{\alpha}$ and represents all the variables on which $\alpha$ depends, i. e., variables needed to describe its abstract states. A *projection* of the planning task over a subset of variables $V \in \mathcal{V}$ is defined by restricting the initial state, goals and preconditions/effects of the operators to $V$. An *atomic abstraction* wrt variable $v$, $\Pi_v$, is the projection of the problem $\Pi$ over that variable. The *synchronized product* of two abstractions $\alpha_1$ and $\alpha_2$ is a standard operation deriving a new state space $\Theta^{\alpha_1 \otimes \alpha_2} = (S', L, T', s'_0, S'_{\star})$ where $S' = S^{\alpha_1} \times S^{\alpha_2}$, $T' = \{((s_1, s_2), l, (s'_1, s'_2)) | (s_1, l, s'_1) \in T^{\alpha_1}$ and $(s_2, l, s'_2) \in T^{\alpha_2}\}$, $s'_0 = (s^{\alpha_1}_0, s^{\alpha_2}_0)$ and $S'_{\star} = \{(s_1, s_2) | s_1 \in S^{\alpha_1}_{\star} \wedge s_2 \in S^{\alpha_2}_{\star}\}$.

*Pattern Databases* (PDBs) derive an abstraction as the projection of the problem over a subset of variables. Then, the abstract state space is traversed by backward Dijkstra's search [Sievers *et al.*, 2012] to precompute the distance from every abstract state to the closest abstract goal state. Selection of good variable subsets has been carefully analyzed [Haslum *et al.*, 2007; Edelkamp, 2007; Kissmann and Edelkamp, 2011]. *Partial PDBs* [Anderson *et al.*, 2007] aim at searching larger abstract state spaces at the expense of not fully traversing them. The backward search is truncated at goal distance $d$ so that the distance of any expanded abstract state is known. States that were not expanded are assigned the next value larger than $d$. If the partial PDB takes into account all the problem variables, it searches a perimeter around the goal [Dillenburg and Nelson, 1994; Manzini, 1995]. Perimeter search has been combined with PDBs for heuristic search domains, by storing the distance to states in the perimeter instead of the distance to the goal state [Felner and Ofek, 2007; Linares López, 2008].

## 2.1 Symbolic Pattern Databases

Binary Decision Diagrams (BDDs) [Bryant, 1986] and Algebraic Decision Diagrams (ADDs) [Bahar *et al.*, 1997] are data structures to efficiently represent boolean and integer functions, respectively. Symbolic search uses BDDs to represent sets of states as their *characteristic functions*. Finite-domain state variables are represented in the BDDs with a logarithmic number of binary variables. In the rest of the paper we assume that all variables are binary without loss of generality. Given a variable ordering, there is a unique reduced representation of every set of states, which may be exponentially smaller than an explicit representation. Variable ordering is decided before starting the search and remains fixed [Kissmann and Edelkamp, 2011]. Moreover, the representation of planning operators as transition relations allows symbolic search to efficiently compute the successor/predecessor sets with the *image/pre-image* operations [Torralba *et al.*, 2013].

*Symbolic PDBs* take advantage of a symbolic representation to succinctly represent the heuristic as an ADD [Ball and Holte, 2008], so that it becomes feasible to explore larger abstract state spaces [Edelkamp, 2005]. The optimal cost of abstract states is determined by a symbolic backward Dijkstra's search. A symbolic Dijkstra's search is characterized as a tuple $(\text{Open}, \text{Closed}, d)$ where Open and Closed are sets of BDDs representing the sets of states that have been reached or expanded and $d$ is the current frontier cost. Open and Closed stand for generated and expanded states, respectively, while *buckets* $\text{Open}_c$ and $\text{Closed}_c$ stand for the sets of states reached or expanded with cost $c$. The backward search is initialized with $\text{Open}_0 = S_{\star}$, $\text{Closed} = \{\varnothing\}$ and $d = 0$. At each step, $\text{Open}_d$ is expanded, removed from Open, inserted in Closed and its successors are inserted in Open. Then $d$ is updated to the minimum cost in Open. The closed list of a symbolic backward search can be used as an admissible heuristic for the original planning task. Since Closed separately stores the set of states with each $h$-value, every state in Closed has a known $h$-value. If the search is completed, states not appearing in Closed are dead-ends ($h = \infty$). If the search is still in progress, the cost of all non-expanded states is set to the next frontier cost.

## 2.2 Merge-and-Shrink

M&S is a flexible way to derive abstractions taking into account all the problem variables. M&S is initialized with the set of atomic abstractions of a planning problem (one for each variable) and iteratively replaces a pair of abstractions by their synchronized product (merge). When the abstract state space is too large to explicitly represent it, M&S reduces it by unifying pairs of states (shrink). The performance of M&S greatly depends on the policies chosen. A *merge policy* decides which two abstractions to merge next. We say it is a *linear merge strategy* if, at each merge step, at least one of the two abstractions to be merged is an atomic abstraction (single variable abstraction). Throughout this paper, a linear merge strategy is assumed since it guarantees an efficient symbolic representation of the heuristic as an Algebraic Decision Diagram [Edelkamp *et al.*, 2012]. Therefore, the merge strategy can be characterized by the ordering by which variables are merged. A *shrinking policy* unifies abstract states, considering them equivalent. Thus, it induces an *equivalence relation*; a mapping from abstract states to buckets. A shrinking policy is *$h$-preserving* wrt a state space $\Theta$ iff it only unifies states with the same goal distance in $\Theta$. Given an abstraction $\alpha$, an equivalence relation is *locally $h$-preserving* iff it is $h$-preserving wrt to its state space $\Theta^{\alpha}$ and *globally $h$-preserving* if it is $h$-preserving wrt $\Theta^{\alpha} \otimes \Pi_{\mathcal{V} \setminus \mathcal{V}_{\alpha}}$ (i. e., the synchronous product of the abstraction and its non-relevant variables). Previous work on M&S mainly focused on defining shrinking policies. DFP[2] strategies unify states with the same $g$ and $h$-values in the abstract state space. Bisimulation [Nissim *et al.*, 2011a] approaches unify abstract states with equivalent transitions (to the same states and with the same labels). Relaxed variants of bisimulation take only into account a transition subset. *Greedy bisimulation* ignores all the transitions going to states further from the goal (i. e., $(s, l, s')$ such that $h_{\alpha}(s) < h_{\alpha}(s')$). Label-catching bisimulation only takes into account transitions labelled with a previous selected set of relevant-labels [Katz *et al.*, 2012]. All the previous strategies are locally $h$-preserving and non-greedy bisimulation is a globally $h$-preserving policy.

[2]Named after Dräger, Finkbeiner and Podelski [Dräger *et al.*, 2006].

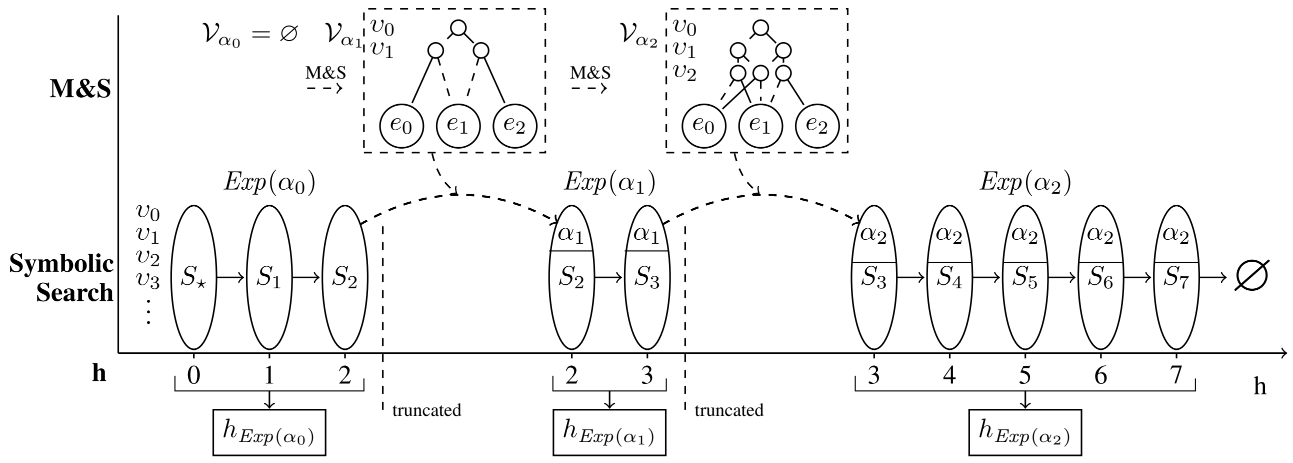
M&S
$\mathcal{V}_{\alpha_0} = \varnothing$
$\mathcal{V}_{\alpha_1}$
$\mathcal{V}_{\alpha_2}$
$v_0$ $v_1$ $v_2$
$e_0$ $e_1$ $e_2$
M&S
Symbolic Search
$v_0$ $v_1$ $v_2$ $v_3$
$Exp(\alpha_0)$ $Exp(\alpha_1)$ $Exp(\alpha_2)$
$S_\star$ $S_1$ $S_2$ $S_3$ $S_4$ $S_5$ $S_6$ $S_7$ $\varnothing$
$\alpha_1$ $\alpha_2$
h
0 1 2 2 3 3 4 5 6 7
truncated
truncated
$h_{Exp(\alpha_0)}$ $h_{Exp(\alpha_1)}$ $h_{Exp(\alpha_2)}$

Figure 1: SM&S example with binary variables, unary cost operators and a limit of six abstract states for M&S.

## 3 Symbolic Merge-and-Shrink

Our proposed technique, SM&S, derives a symbolic heuristic using M&S abstractions to relax a symbolic backward search. Figure 1 represents a high level view of the interaction between symbolic search and M&S abstractions in the SM&S algorithm. SM&S starts computing a symbolic perimeter, $Exp(\alpha_0)$. The first BDD with $h = 0$ contains the goal states and by successive *pre-image* operations SM&S generates the sets of states with $h = 1$, $h = 2$, etc. This search is intractable for general planning domains so when memory or time bounds are surpassed, it is truncated ($h = 2$ in Figure 1). The minimum distance to the goal of the expanded state sets is stored, transforming the list of BDDs representing the search to an ADD representing the heuristic, $h_{Exp(\alpha_0)}$.

Then, M&S is used to derive an abstraction, $\alpha_1$. M&S merges variables, applying shrinking if needed to fit the maximum number of abstract states (in Figure 1, $\alpha_1$ must have at most three abstract states before merging the next variable). SM&S uses $\alpha_1$ to relax the top levels of all the BDDs in the exploration $Exp(\alpha_1)$. All partial states related to the same abstract state are considered equivalent so that when one is reached, all of them are. In Figure 1, abstract state $e_1$ represents partial states 00 and 10. During the exploration, if state $10010\ldots$ is reached, then state $00010\ldots$ is also reached and vice versa. Hence, BDD nodes pointed to by 00 and 10 are equivalent, making the top part of any BDD in the exploration equal to its M&S representation. Also, M&S abstractions are cumulative, so the top levels of $\alpha_2$ coincide with those of previous abstractions. SM&S continues interleaving symbolic explorations and M&S iterations until an exploration is completed or time/memory bounds are violated. When finished, it returns the list of ADDs representing the heuristic.

Algorithm 1 shows the SM&S algorithm. It receives as input a planning task $\Pi$ and some parameters to bound the memory and time resources. The output is a heuristic $H$ represented as a list of ADDs. Each ADD is the result of a backward symbolic exploration over a M&S abstraction $\alpha$. If SM&S reaches the time limit $T_{\text{SM\&S}}$ a last ADD is included (line 16). This last ADD represents the minimum cost from each abstract state to the closest abstract goal state, computed as the original M&S heuristic, with an explicit traversal of the abstract state space. SM&S starts initializing the symbolic backward search as usual to $(\text{Open} = S_\star, \text{Closed} = \varnothing, d = 0)$ and $\alpha$ is empty. Symbolic search progresses following the relaxation imposed by $\alpha$ (line 7).

**Definition 1.** *Let $\alpha$ be an abstraction with relevant variables $\mathcal{V}_\alpha$ and a subset of non-relevant variables $\mathcal{V}' \subseteq \mathcal{V} \setminus \mathcal{V}_\alpha$. A $\mathcal{V}'$* symbolic exploration *on $\alpha$, $Exp(\alpha, \mathcal{V}')$, is a backward search over the synchronized product of $\alpha$ and the projection of the planning task over $\mathcal{V}'$: $\alpha^{\mathcal{V}'} = \alpha \otimes \Pi_{\mathcal{V}'}$. When all the non-relevant variables are considered in the exploration, $\mathcal{V}' = \mathcal{V} \setminus \mathcal{V}_\alpha$, we say it is a* full symbolic exploration. *When $\mathcal{V}' = \varnothing$, the exploration searches the abstract state space, $\Theta^\alpha$.*

SM&S uses full symbolic explorations, always considering all the non-relevant variables for the abstraction. In the rest of the paper any mention of a symbolic exploration $Exp(\alpha)$ stands for a full symbolic exploration $Exp(\alpha, \mathcal{V} \setminus \mathcal{V}_\alpha)$. The `Explore` procedure performs a full symbolic exploration of $\alpha$, updating the open and closed lists and the frontier cost, $d$. At each step, the current frontier $S_e$ is extracted from $\text{Open}_d$, removing already expanded states (line 20). Before applying non-zero cost operators, a breadth-first search applying only 0-cost operators is performed in order to obtain all states reachable with cost $d$ (lines 23 - 25). Those states are stored in the closed list and their successors are generated and inserted in the corresponding bucket of the open list. $pre\text{-}image_c$ computes the set of predecessor states in the state space $\alpha \otimes \Pi_{\mathcal{V} \setminus \mathcal{V}_\alpha}$ with operators of cost $c$. The exploration finishes when the open list is empty or the search is truncated. The heuristic derived from its closed list is stored in $H$ as an ADD (line 8).

**Definition 2.** *Given a symbolic exploration $Exp(\alpha)$ whose contents are $\{Open, Closed, d\}$. Let $b(Open) > d$ be the cost of the next bucket in Open, and the next frontier cost be $d' = \min\{b(Open), d + \min_{o \in \mathcal{O}} c(o)\}$, the minimum between the next bucket in the open list and the current frontier plus the cost of the cheapest operator. The exploration heuristic is defined as in partial PDBs, setting not expanded states either*

**Algorithm 1:** Symbolic Merge-and-Shrink.

```
   Input: Task Π, Memory bounds: N, $N_F$
     Time bounds: $T_{SM\&S}, T_{Sym}, T_{Exp}, T_I$
   Output: List of ADDs: H
1  $H \leftarrow \varnothing$
2  $abs \leftarrow \{\pi_v | v \in \mathcal{V}\}$
3  $\alpha \leftarrow \varnothing$
4  $(\text{Open}, \text{Closed}, d) \leftarrow (s_\star, \varnothing, 0)$
5  while $Open \neq \varnothing$ and $t_s < T_{SM\&S}$:
6      if $|Open_d| \leq N_F$ and $t_s < T_{Sym}$:
7          Explore($\alpha$, Open, Closed, $d$, $N_F$, $T_{Exp}$, $T_I$)
8          $H \leftarrow H \cup$ ADD(Closed, $d$)
9      Select $\pi_v \in abs$
10     $abs \leftarrow abs \setminus \pi_v$
11     $E \leftarrow$ Shrink($\alpha, \frac{N}{size(\pi_v)}$)
12     Open $\leftarrow$ Open$^{\exists E}$
13     Closed $\leftarrow$ Closed$^{\forall E}$
14     $\alpha \leftarrow \alpha^E \otimes \pi_v$ // Merge next variable
15 if $Open \neq \varnothing$:
16     $H \leftarrow H \cup$ Explicit-Search($\alpha$)
17 return $H$

18 Procedure Explore($\alpha$, Open, Closed, $d$, $N_F$, $T_{Exp}$, $T_I$)
19 while $Open \neq \varnothing$ and $t_{exp} < T_{Exp}$:
20     $S_e \leftarrow \text{Open}_d \wedge \neg\text{Closed}$
21     $\text{Open}_d \leftarrow \varnothing$
22     $\text{Closed}_d \leftarrow S_e$
       // Breadth First Search with 0-cost operators
23     while $S_e \neq \varnothing$ and $|S_e| < N_F$ and $t < T_{Exp}$:
24         $S_e \leftarrow pre\text{-}image_0(S_e, \alpha, T_I) \wedge \neg\text{Closed}$
25         $\text{Closed}_d \leftarrow \text{Closed}_d \vee S_e$
       // Apply cost operators
26     if $|Closed_d| < N_F$ and $t_{exp} < T_{Exp}$:
27         for $c = 1, \ldots, C$:
28             $\text{Open}_{d+c} \leftarrow \text{Open}_{d+c} \vee$
29                 $pre\text{-}image_c(\text{Closed}_d, \alpha, T_I)$
30         $d \leftarrow \min_c \text{Open}_c \neq \varnothing$
```

*to $d'$ or $\infty$, depending on whether the exploration successfully finished or not:*

$$h_{Exp(\alpha)}(s) = \begin{cases} c & s \in \mathit{Closed}_c \\ d' & s \notin \mathit{Closed} \text{ and } \mathit{Open} \neq \varnothing \\ \infty & s \notin \mathit{Closed} \text{ and } \mathit{Open} = \varnothing \end{cases}$$

Several parameters bound the memory and time used by the algorithm. Memory is controlled by the maximum number of M&S abstract states $N$ and the maximum number of nodes $N_F$ to represent the search frontier. Four different parameters limit the time spent by the algorithm $t_s$ or by the exploration $t_{exp}$. $T_{\text{M\&S}}$ aborts the heuristic generation to guarantee termination. $T_{Sym}$ prevents SM&S from performing more symbolic explorations to focus on completing the M&S abstraction. $T_{Exp}$ fixes a maximum time for each individual exploration to avoid consuming all the time in one single exploration. Finally, $T_I$ limits the maximum allotted time in one $pre\text{-}image$. If $T_I$ is exceeded, not only the $image$ but the whole exploration is halted. In order to avoid starting another $image$ as hard as the halted one, the maximum number of nodes in the frontier search is reduced to $N_F = \frac{|S_e|}{2}$. All these parameters are set independently of the domain and only depend on the memory and time resources available to the planner.

After an exploration is truncated, abstraction $\alpha$ is shrunk in order to merge a new variable. Instead of restarting the search from scratch in the new abstract state space, the new exploration continues from the current frontier, relaxing the open and closed lists. That way, the exploration is more informed since the first steps were performed on less-relaxed state spaces. Besides, it prevents SM&S from starting a new exploration per each variable merged, which could cause redundant work in case that most searches were truncated at the same frontier cost. SM&S keeps including variables into the abstraction until the search has been simplified enough, i. e., the relaxed frontier is smaller than the parameter $N_F$. We define existential and universal shrinking wrt equivalence relation $E$, to perform the relaxation over the open and closed lists, respectively. The new exploration is initialized to $(\text{Open}^{\exists E}, \text{Closed}^{\forall E}, d)$. To avoid pruning valid states, the relaxed open list must keep all the states and no states can be added to the closed list so that the following inequality holds for any state set $S_B$: $S_B^{\forall E} \subseteq S_B \subseteq S_B^{\exists E}$.

**Definition 3.** *Let $S_B$ be a state set and $E$ an equivalence relation over $\mathcal{V}_\alpha \subset \mathcal{V}$. Each bucket $e \in E$ represents a set of equivalent abstract states $s_e$, and it is described as their disjunction: $e = \bigvee_{s \in s_e} s$. Existential/universal shrinking of $S_B$ wrt $E$ is defined as:*

$$S^{\exists E} = \bigvee_{e \in E} \big(\exists \mathcal{V}_\alpha (S_B \wedge e)\big) \wedge e$$
$$S^{\forall E} = \bigvee_{e \in E} \big(\forall \mathcal{V}_\alpha (S_B \wedge e) \vee \overline{e}\big) \wedge e$$

For every bucket $e$, $S_B \wedge e$ is the subset of $S_B$, which corresponds to any abstract state in $e$. Then, Existential/Universal quantification is a standard BDD operation which removes variables $\mathcal{V}_\alpha$ keeping the values of other variables reached for any/every abstract state in $e$, respectively. Finally, the value of $\mathcal{V}_\alpha$ is set to $e$. Disjunction with $\overline{e}$ is necessary to ignore other buckets in the universal quantification. After relaxing the exploration, the search continues over the new abstract state space so the $pre\text{-}image$ generates predecessor states in the new abstract state space. To avoid recomputing the transition relation for each exploration, we compute the $pre\text{-}image$ with the transition relation of the original problem and then apply existential shrinking to relax the result. Existential shrinking is not the computational bottleneck, though transformations of the transition relation could optimize $image$ computation.

Once the heuristic is computed as a list of ADDs, it can be used in the search to solve the planning problem. One must be careful to preserve admissibility when combining heuristic estimates of ADDs coming from different explorations. In particular, if a state was expanded by an exploration, latter explorations may return inadmissible estimations for it. When relaxing the closed list, the state may be removed by universal shrinking and re-expanded with higher cost in latter explorations. Thus, the heuristic is computed as the maximum estimation of each exploration, but ignoring those explorations after the first one that expanded the state. Let $s$ be a state, and $I$ be the first exploration in which the $s$ was expanded, $I = min\{i | s \in \text{Closed}_{Exp(\alpha_i)}\}$. Then, we define $h_{\text{SM\&S}}(s) = max_{i=0,I} h_{Exp(\alpha_i)}(s)$.

### 3.1 Theoretical Properties

**Theorem 1.** *$h_{SM\&S}$ heuristic is admissible and consistent.*

**Proof.** $h$ is consistent iff $h(s) \leq h(s') + c(l)\ \forall (s, l, s')$. Given the definition of $h_{\text{SM\&S}}$, (1) if $s'$ was not expanded in any exploration, then its $h$-value equals $\infty$ or the larger frontier cost. In either case, $h_{\text{SM\&S}}(s) \leq h_{\text{SM\&S}}(s')\ \forall s$. (2) Otherwise, let $Exp(\alpha_i)$ be the first exploration expanding $s'$, and doing so with cost $c$. Then $h_{\text{SM\&S}}(s') \geq c$. As $s'$ was expanded, necessarily $s$ was inserted in $\text{Open}_{c+c(l)}$. $h_{\text{SM\&S}}(s) \leq c + c(l)$ holds if $\text{Open}_{c+c(l)}$ is expanded by any exploration, since $s$ remains in Open (or has been expanded with a lower cost) because existential shrinking preserves all the states in the open list and universal shrinking prevents states from being closed if they are not expanded. If SM&S finishes before reaching $c+c(l)$ (the search is truncated), $h(s)$ will be the next frontier cost $d' \leq c + c(l)$.

Admissibility is derived from consistency, given that the heuristic is perfect for goal states: $\forall s_\star \in S_\star,\ h(s_\star) = 0$. □

One of the advantages of M&S is that it is possible to control its time and memory usage by appropriately setting the maximum number of abstract states, $N$. Assuming that $N$ is polynomially bounded, computing the abstraction has polynomial complexity [Helmert *et al.*, 2007]. SM&S does not ensure that the full symbolic exploration over an abstraction has a more concise BDD representation than the original problem. Also, relaxing the search frontier with existential shrinking could actually enlarge it. Fortunately, we can derive a bound on the maximum number of BDD nodes needed to represent any set of states in the exploration.

**Theorem 2.** *Let $\alpha$ be a M&S abstraction with relevant variables $\mathcal{V}_\alpha$, generated with a maximum number of abstract states $N$. Let $S_B$ be a BDD describing a set of states on $\Theta^\alpha \otimes \Pi_{\mathcal{V} \setminus \mathcal{V}_\alpha}$ using a variable order whose first/top variables correspond to $\mathcal{V}_\alpha$. Then, the size of $S_B$ is bounded by:*

$$|S_B| \leq N \left( |\mathcal{V}_\alpha| + 2^{|\mathcal{V} \setminus \mathcal{V}_\alpha| + 1} \right)$$

**Proof.** $S_B$ may be divided in its top and bottom parts. Each top layer corresponds to an intermediate M&S abstraction, with one node per abstract state. As M&S imposes the limit on the number of abstract states for every intermediate abstraction, each layer has at most $N$ nodes. Thus, the number of nodes in the top part of $S_B$ is bound by $N|\mathcal{V}_\alpha|$. The bottom layers correspond to functions over $\mathcal{V} \setminus \mathcal{V}_\alpha$ describing which values are reachable for each abstract state. Each of these functions is described as a BDD with $|\mathcal{V} \setminus \mathcal{V}_\alpha|$ levels. In the worst case, they do not share any node, and each one has $2^{|\mathcal{V} \setminus \mathcal{V}_\alpha|+1}$ nodes. Since there are $N$ different functions, the size of the bottom part of $S_B$ is bound by $N2^{|\mathcal{V} \setminus \mathcal{V}_\alpha|+1}$. □

As variables are merged into the abstraction, the complexity of its full symbolic exploration decreases. The bound on the top part of the BDD grows linearly on $N$, while the bound on its bottom part is exponentially reduced. Thus, eventually the full symbolic exploration of the abstraction will be tractable. In the limit, the exploration is performed over a linear sized state space and can be explicitly explored, just as the original M&S algorithm does. Theorem 2 requires relevant variables for the abstraction to be placed in the top levels of exploration BDDs. For this to hold in every symbolic exploration, the symbolic search must use the same variable ordering as the M&S merge strategy. Since changing the variable ordering of a BDD may cause an exponential blow-up, we use the same variable ordering in order to guarantee an efficient computation of SM&S.

## 4 Experimental Evaluation

We compare the performance of SM&S heuristics against the two methods it combines: M&S and symbolic perimeter search (SP). The heuristics are implemented in the FAST DOWNWARD planning system (FD) [Helmert, 2006]. BDD operations are implemented in Fabio Somenzi's CUDD library 2.5.0.[3] Experiments were run and validated with the IPC-2011 software on a single core of an Intel Xeon X3470 processor at 2.93 GHz. We compare the coverage of A* search, as implemented in FD, with every heuristic. The experimental setting is the same as in IPC-2011: 1800 seconds per problem and 6 GB of available memory. SM&S parameters were manually set to fit the competition setting: the maximum number of nodes to represent the state set to expand is $N_F = 10{,}000{,}000$. A maximum time is set to each individual $image$ ($T_I = 30s$), exploration ($T_{Exp} = 300s$) and symbolic search ($T_{Sym} = 900s$). Both SP and SM&S take 1200 seconds to generate the heuristic ($T_{SM\&S} = 1200s$).

We use the two shrinking strategies used by the M&S planner entering the competition [Nissim *et al.*, 2011b]: bisimulation (bop) with $N = 200{,}000$ and greedy bisimulation (gop) with $N = \infty$. After applying the bisimulation criterion, if the number of abstract states is greater than $N$, a DFP strategy is applied. We also use the same strategies with a more restrictive threshold of $N = 10{,}000$ to avoid memory exhaustion while generating the heuristic. Also, in order to analyze the relevance of shrinking strategies, we introduce SM&S-all, a naïve variant of shrink which considers equivalent all abstract states ($N = 1$). Thus, SM&S-all abstracts away all the variables in the abstraction, and explorations are just symbolic PDBs ignoring the top variables.

Variable ordering has a relevant impact on the performance of both M&S and symbolic search. There are two proposed variable ordering optimization criteria by the planners FD and GAMER. FD places variables before those that depend on them, so that the goal variables are usually placed at the end. GAMER ordering attempts to put together variables that depend on each other, so that goal variables are usually placed in the middle. No order dominates the other. Indeed, they have been shown to work best for different domains and techniques. FD ordering seems better for M&S

[3]http://vlsi.colorado.edu/~fabio/CUDD

| | FAST DOWNWARD Ordering | | | | | | | | | | GAMER Ordering | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | M&S | | | | SM&S | | | | | | M&S | | | | SM&S | | | | | |
| | bop | | gop | | bop | | gop | | all | SP | bop | | gop | | bop | | gop | | all | SP |
| | 10k | 200k | 10k | $\infty$ | 10k | 200k | 10k | $\infty$ | | | 10k | 200k | 10k | $\infty$ | 10k | 200k | 10k | $\infty$ | | |
| BARMAN | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| ELEVATORS | 9 | 12 | 9 | 0 | 18 | 18 | 18 | 2 | 18 | 18 | 12 | 14 | 12 | 4 | **19** | **19** | **19** | 4 | 18 | **19** |
| FLOORTILE | 4 | 7 | 3 | 3 | **12** | **12** | **12** | **12** | **12** | **12** | 2 | 3 | 2 | 3 | **12** | **12** | **12** | 10 | **12** | **12** |
| NOMYSTERY | 18 | **19** | 13 | 14 | 18 | 17 | 16 | 16 | 14 | 14 | 14 | 16 | 14 | 14 | 15 | 16 | 14 | 16 | 14 | 14 |
| OPENSTACKS | **16** | **16** | **16** | 4 | 15 | 15 | 15 | 4 | 15 | 4 | **16** | **16** | **16** | 4 | **16** | **16** | **16** | 15 | **16** | 15 |
| PARC-PRINTER | 12 | **13** | 12 | 11 | 12 | **13** | 12 | 11 | 9 | 7 | 12 | 12 | 12 | 11 | 12 | 12 | 12 | 11 | 8 | 7 |
| PARKING | 3 | 0 | **7** | **7** | 6 | 0 | **7** | **7** | **7** | 0 | 6 | 0 | 1 | 0 | 6 | 0 | 1 | 0 | 1 | 0 |
| PEG-SOLITAIRE | 19 | 19 | 19 | 0 | 19 | 19 | **20** | 0 | 19 | 19 | 17 | 19 | 17 | 0 | 19 | 19 | 19 | 0 | 19 | 19 |
| SCANALYZER | 9 | **11** | 10 | 6 | 9 | 9 | 10 | 9 | 10 | 9 | 9 | 6 | 9 | 3 | 9 | 9 | 9 | 9 | 9 | 9 |
| SOKOBAN | 19 | 19 | 19 | 1 | 19 | 19 | 19 | 1 | 18 | 10 | **20** | **20** | **20** | 3 | **20** | 16 | **20** | 7 | 19 | 16 |
| TIDYBOT | 12 | 0 | 12 | **13** | 12 | 0 | 12 | **13** | 12 | 8 | 11 | 0 | 6 | **13** | 12 | 0 | 6 | **13** | 10 | 8 |
| TRANSPORT | 6 | 7 | 6 | 6 | 7 | 8 | 7 | 7 | 7 | 7 | 6 | 6 | 6 | 6 | 8 | 8 | 8 | 8 | **9** | 8 |
| VISITALL | 9 | 9 | 9 | 16 | 12 | 12 | 12 | **17** | 12 | 12 | 10 | 9 | 10 | 11 | 13 | 10 | 13 | 12 | 12 | 12 |
| WOODWORKING | 6 | 6 | 9 | 9 | 7 | 6 | 9 | 9 | 7 | 6 | 7 | 8 | 9 | 9 | 12 | 12 | **13** | **13** | 11 | 12 |
| TOTAL | 146 | 142 | 148 | 94 | 170 | 152 | 173 | 112 | 164 | 130 | 146 | 133 | 138 | 85 | **177** | 153 | 166 | 122 | 162 | 155 |

Table 1: Coverage results for problems of the sequential optimal track of IPC-2011.

while GAMER ordering produces better results with symbolic search [Edelkamp *et al.*, 2012]. We report experiments with both variable orderings to see their impact on SM&S.

Table 1 shows coverage results of all the planner configurations in IPC-2011 domains, using the IPC problem set with 20 problems per domain. Results show that SM&S successfully combines the two techniques, obtaining a heuristic usually as good as the best of them. In some domains there is a synergy between the M&S relaxation and the symbolic search. Besides, SM&S gradually relaxes the search frontier, helping to decide how many variables should be relaxed. Remarkably, different configurations of SM&S-gop are able to solve more problems in PEG-SOLITAIRE (20) and VISITALL (17) than all the other versions and IPC-2011 planners. Overall, the best SM&S configuration solves 177 problems, using GAMER ordering and bisimulation shrinking with $N = 10{,}000$. The few cases where SM&S solves less problems are due to memory failures caused by the symbolic search overhead. SM&S-all results show that relaxing the perimeter search is helpful in some domains, especially OPENSTACKS and PARKING. However, SM&S-all ignores all variables in the abstraction. This can be improved by using a better shrinking policy. Bisimulation shrinking does not help much when relaxing the frontier search, since bisimilar abstract states are always represented with the same nodes in the BDDs. Thus, improvement from SM&S-all to SM&S-bop can be attributed to the DFP strategy (bisimulation might help to merge more variables before applying DFP shrinking). Finally, greedy bisimulation helps in some domains, such as PEG-SOLITAIRE or WOODWORKING, allowing SM&S to obtain more solutions. Still, the gap between different shrinking versions is not too large in most domains. The comparison of variable orderings in SM&S-all gives more insights about why FD ordering is better for M&S abstractions. Even though GAMER ordering is better for SP, FD ordering has an advantage when relaxing the top variables. The reason is that most relevant variables (goal variables) are placed at the bottom so they are not affected by shrinking.

Heuristic generation time is also relevant for the heuristic comparison. In our setting, SM&S takes more time to generate the heuristic than M&S and the latter does not take advantage of the extra time in the search, since A$^*$ rapidly exhausts the available memory when using a precomputed heuristic such as PDBs, M&S or SM&S. Hence, SM&S generates better heuristics at the expense of taking more time. Still, most of the improvement from M&S to SM&S can be attributed to the combination of symbolic search and M&S and not to the extra time used for computing the heuristic.

## 5 Conclusions

In this work we introduced SM&S, a novel combination of symbolic search and M&S abstractions to generate better admissible heuristics for optimal planning. SM&S uses M&S abstractions to relax a symbolic backward search, keeping only partial information about some variables of the problem. Experimental results show that SM&S successfully obtains results at least as good as the best of either one of the two techniques it combines. Even though M&S abstractions are highly dependent of the shrinking policy, SM&S improves the performance of a symbolic perimeter even with a simple shrinking strategy, SM&S-all. Moreover, it improves M&S results for all shrinking policies used. On the other hand, bisimulation shrinking derives good abstractions for SM&S in some domains but not in others. This suggests there is still room for improvement with new shrinking policies specific for SM&S (i. e., taking into account the symbolic search frontier and not preserving information about states already expanded by previous explorations).

## Acknowledgments

This work has been partially supported by MICINN projects TIN2008-06701-C03-03 and TIN2011-27652-C03-02.

## References

[Anderson *et al.*, 2007] Keith Anderson, Jonathan Schaeffer, and Rob C. Holte. Partial pattern databases. In *SARA*, pages 20–34, 2007.

[Bahar *et al.*, 1997] R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebric decision diagrams and their applications. *Form. Methods Syst. Des.*, 10(2-3):171–206, April 1997.

[Ball and Holte, 2008] Marcel Ball and Robert C. Holte. The compression power of symbolic pattern databases. In *ICAPS*, pages 2–11, 2008.

[Bryant, 1986] R.E. Bryant. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on*, C-35(8):677 –691, aug. 1986.

[Culberson and Schaeffer, 1998] Joseph C. Culberson and Jonathan Schaeffer. Pattern databases. *Computational Intelligence*, 14(3):318–334, 1998.

[Dillenburg and Nelson, 1994] John F. Dillenburg and Peter C. Nelson. Perimeter search. *Artif. Intell.*, 65(1):165–178, 1994.

[Dräger *et al.*, 2006] Klaus Dräger, Bernd Finkbeiner, and Andreas Podelski. Directed model checking with distance-preserving abstractions. In *Model Checking Software*, pages 19–34. Springer, 2006.

[Edelkamp *et al.*, 2012] Stefan Edelkamp, Peter Kissmann, and Álvaro Torralba. Symbolic A$^*$ search with pattern databases and the merge-and-shrink abstraction. In *ECAI*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 306–311, 2012.

[Edelkamp, 2001] Stefan Edelkamp. Planning with pattern databases. In *ICAPS*, pages 13–34, 2001.

[Edelkamp, 2005] Stefan Edelkamp. External symbolic heuristic search with pattern databases. In *ICAPS*, pages 51–60, 2005.

[Edelkamp, 2007] Stefan Edelkamp. Automated creation of pattern database search heuristics. In *MOCHART*, 2007.

[Felner and Ofek, 2007] A. Felner and N. Ofek. Combining perimeter search and pattern database abstractions. In *SARA*, pages 414–429, 2007.

[Haslum *et al.*, 2007] Patrick Haslum, Adi Botea, Malte Helmert, Blai Bonet, and Sven Koenig. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *AAAI*, pages 1007–1012, 2007.

[Helmert *et al.*, 2007] Malte Helmert, Patrik Haslum, and Jörg Hoffmann. Flexible abstraction heuristics for optimal sequential planning. In *ICAPS*, pages 176–183, 2007.

[Helmert *et al.*, 2011] Malte Helmert, Gabriele Röger, Jendrik Seipp, Erez Karpas, Jörg Hoffmann, Emil Keyder, Raz Nissim, Silvia Richter, and Matthias Westphal. Fast downward stone soup. IPC-2011 planner abstracts, 2011.

[Helmert, 2006] Malte Helmert. The fast downward planning system. *JAIR*, 26:191–246, 2006.

[Katz *et al.*, 2012] Michael Katz, Jörg Hoffmann, and Malte Helmert. How to relax a bisimulation? In *ICAPS*, 2012.

[Kissmann and Edelkamp, 2011] Peter Kissmann and Stefan Edelkamp. Improving cost-optimal domain-independent symbolic planning. In *AAAI*, pages 992–997, 2011.

[Linares López, 2008] Carlos Linares López. Multi-valued pattern databases. In *ECAI*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 540–544, 2008.

[Manzini, 1995] Giovanni Manzini. BIDA$^*$: an improved perimeter search algorithm. *Artificial Intelligence*, 75:347–360, 1995.

[Nissim *et al.*, 2011a] Raz Nissim, Jörg Hoffmann, and Malte Helmert. Computing perfect heuristics in polynomial time: On bisimulation and merge-and-shrink abstraction in optimal planning. In *IJCAI*, pages 1983–1990, 2011.

[Nissim *et al.*, 2011b] Raz Nissim, Jörg Hoffmann, and Malte Helmert. The merge-and-shrink planner: Bisimulation-based abstraction for optimal planning, 2011.

[Sievers *et al.*, 2012] Silvan Sievers, Manuela Ortlieb, and Malte Helmert. Efficient implementation of pattern database heuristics for classical planning. In *SOCS*, 2012.

[Torralba *et al.*, 2013] Álvaro Torralba, Stefan Edelkamp, and Peter Kissmann. Transition trees for cost-optimal symbolic planning. In *ICAPS*, 2013.