Focusing on What Really Matters: Irrelevance Pruning in Merge-and-Shrink

Álvaro Torralba and **Peter Kissmann**
Saarland University
Saarbrücken, Germany
torralba@cs.uni-saarland.de, kissmann@cs.uni-saarland.de

Abstract

Merge-and-shrink (M&S) is a framework to generate abstraction heuristics for cost-optimal planning. A recent approach computes simulation relations on a set of M&S abstractions in order to identify states that are better than others. This relation is then used for pruning states in the search when a "better" state is already known. We propose the usage of simulation relations inside the M&S framework in order to detect irrelevant transitions in abstract state spaces. This potentially simplifies the abstraction allowing M&S to derive more informed heuristics. We also tailor M&S to remove irrelevant operators from the planning task. Experimental results show the potential of our approach to construct well-informed heuristics and simplify the planning tasks prior to the search.

Introduction

Dominance pruning methods define a relation between states in the state space that determines which states are better than others. Whenever a state in the search is dominated by another previously generated state, it can be pruned (Hall et al. 2013). Recently, a new approach to derive admissible dominance relations in a domain-independent way has been proposed (Torralba and Hoffmann 2015). It uses merge-and-shrink (M&S) in order to construct a set of abstractions and then compute a simulation relation on the abstract state spaces. The individual simulation relations can be combined to derive a dominance relation between states that allows to prune the search in an admissible way. Dominance pruning significantly reduces search effort in a number of domains, at the price of incurring in a significant overhead for each node due to the comparison against previously known states.

While previous work only used M&S prior to computing a simulation relation, in this paper we make use of the simulation relations inside the M&S algorithm in two different ways. On the one hand, we define subsumed transition pruning, a safe method to prune irrelevant transitions in M&S abstractions. On the other hand, we define similarity shrinking, a shrinking strategy that derives the perfect heuristic while potentially reducing the abstraction size with respect to bisimulation. Experimental results show how M&S heuristics may benefit from these techniques.

Moreover, we tailor the M&S framework for pruning irrelevant operators, aiming to reduce the accidental complexity of planning tasks (Haslum 2007). Compared to dominance pruning, an important advantage is that the operators are removed from the task before starting the search, avoiding any overhead for node expansions. Removed operators are not considered by the heuristics or other search enhancements, potentially improving their estimations and speeding up its computation. Finally, while dominance pruning is more tightly related to A*search, our irrelevance pruning can be used in other algorithms, such as symbolic search (McMillan 1993; Kissmann 2012).

While previous techniques in M&S could have already been used to remove operators that are never applicable or only lead to dead-end states, our subsumed transition pruning detects operators that belong to valid plans but can nevertheless be removed preserving an optimal solution. We compare our results against other pruning techniques that simplify the task by removing operators in a preprocessing phase, using state-invariant constraints (Alcázar and Torralba 2015) or relevant-path analysis (Haslum, Helmert, and Jonsson 2013). The results show that simulation relations can be used in M&S to remove irrelevant operators and improve the performance of state-of-the-art optimal planners.

Background

In this paper we consider classical planning in a finite-domain representation (FDR). An FDR planning task Π is a quadruple $\Pi = \langle \mathcal{V}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$, where $\mathcal{V}$ is a set of variables and each $v \in \mathcal{V}$ is associated with a finite domain $\mathcal{D}_v$. A state is an assignment to all variables, while a partial state is an assignment to a subset of $\mathcal{V}$. For a partial state s we denote the set of variables with assigned values by $V(s)$. $\mathcal{O}$ is the set of operators, each $o \in \mathcal{O}$ being a triple $\langle pre_o, \mathit{eff}_o, c(o) \rangle$, where pre_o and eff_o denote the precondition and the effect of the operator, respectively, given in form of partial states, and $c(o) \in \mathbb{R}_0^+$ is its cost; $\mathcal{I}$ is the initial state, and $\mathcal{G}$ the goal states, the latter in the form of a partial state.

The semantics of such an FDR planning task is given by a labeled transition system (LTS) Θ, which is a tuple $\Theta = \langle \mathcal{S}, \mathcal{L}, \mathcal{T}, \mathcal{I}, \mathcal{S}^G \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{L}$ a set of labels corresponding to the operators $\mathcal{O}$, $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{L} \times \mathcal{S}$ the set of transitions with $(s, l, t) \in \mathcal{T}$ if $pre_l \subseteq s$, $\mathit{eff}_l \subseteq t$ and $t[v] = s[v]$ for all $v \in \mathcal{V} \setminus V(\mathit{eff}_l)$. We alternatively

denote such transitions by $s \xrightarrow{l} t$. $\mathcal{I} \in \mathcal{S}$ is the initial state and $\mathcal{S}^G \subseteq \mathcal{S}$ the set of goal states with $s_g \in \mathcal{S}^G$ iff $\mathcal{G} \subseteq s_g$.

A plan π is a sequence of operators $\pi = \langle o_1, \ldots, o_n \rangle$, and their successive application in $\mathcal{I}$ leads to a goal state. The cost of π is the sum of the costs of the operators in π, i. e., $c(\pi) = \sum_{i=1}^{n} c(o_i)$. A plan is optimal if no plan of smaller cost exists and strongly optimal if its number of 0-cost actions is minimal among all optimal plans. We denote by $h^*(s)$ and $h^{0*}(s)$ to the cost and number of 0-cost actions in a strong optimal plan for s, respectively.

Merge-and-Shrink

Merge-and-shrink (M&S) is an algorithm initially devised in the model-checking area (Dräger, Finkbeiner, and Podelski 2006; 2009) and later brought to planning to generate admissible abstraction heuristics (Helmert, Haslum, and Hoffmann 2007; Helmert et al. 2014). Here, we consider the formalization by Sievers, Wehrle, and Helmert (2014), where we have a set of LTSs, $X = \{\Theta_1, \ldots, \Theta_k\}$, under the same set of labels, $\mathcal{L}$. X is initialized with the atomic projections of the planning task, i. e., one for each variable. Then, several transformations are available:

- Merge: Replace Θ_i and Θ_j by their synchronized product, $\Theta_i \otimes \Theta_j = \langle S_i \times \mathcal{S}_j, \mathcal{L}, \mathcal{T}^{\otimes}, \mathcal{S}_i^G \times \mathcal{S}_j^G \rangle$, where $(s_i s_j) \xrightarrow{l} (t_i t_j) \in \mathcal{T}^{\otimes}$ iff $s_i \xrightarrow{l} t_i \in \mathcal{T}_i$ and $s_j \xrightarrow{l} t_j \in \mathcal{T}_j$.
- Shrink: Replace Θ_i by an abstraction of Θ_i.

The *global transition system* is implicitly defined as the synchronized product of all the LTSs, $\bigotimes_i \Theta_i$. In our notation, states in the global LTS are denoted as s or t, while we refer to states of each Θ_i as s_i, t_i, etc. At the beginning the global LTS is the state space of the planning task, and after shrinking it is an abstraction. Bisimulation shrinking (Nissim, Hoffmann, and Helmert 2011) is popular, because it is **globally h-preserving**, i. e., preserves the goal-distance of every state in the global LTS. Thus, M&S with bisimulation shrinking derives the perfect heuristic, potentially reducing the size of the state space by an exponential factor.

In the initial atomic abstractions of M&S, the labels $\mathcal{L}$ correspond to the set of operators of the planning task, $\mathcal{O}$. Label reduction aggregates some labels in order to improve bisimulation and further reduce the size of the abstractions. When using label reduction, each label l corresponds to a set of operators, $\mathcal{O}_l$, that contains the operators of all the labels that were aggregated into l. Sievers, Wehrle, and Helmert (2014) identified the conditions under which label reduction is **exact**, i. e., which labels may be aggregated without modifying the global LTS, Θ. Two labels l, l' can be reduced iff $c(l) = c(l')$ and either l is equivalent to l' in all Θ_i except one or l subsumes l' or l' subsumes l in every LTS, Θ_i.

Another common operation in M&S is to remove all unreachable or dead-end states from each Θ_i, since they correspond to unreachable and dead-end states in the global LTS.

Simulation-based Dominance Pruning

In dominance pruning, we define a relation, $\preceq : \mathcal{S} \times \mathcal{S}$, such that $s \preceq t$ implies that for any plan for s an at least as good plan exists for t, so $h^*(t) \leq h^*(s)$ and, if $h^*(t) = h^*(s)$, then $h^{0*}(t) \leq h^{0*}(s)$. For example, consider the position of a package p in a logistics task where a truck must carry several packages to location G. All other state variables having equal values, the best is to have p at G, and it is better for p to be in the truck than at any location other than G.

Simulation relations are commonly used in model-checking to compare two systems (Milner 1971; Grumberg and Long 1994; Loiseaux et al. 1995). A relation $\preceq$: $\mathcal{S} \times \mathcal{S}$ is a simulation for an LTS, Θ, if whenever $s \preceq t$ (in words: t **simulates** s), for every transition $s \xrightarrow{l} s'$ there exists a transition $t \xrightarrow{l} t'$ s.t. $s' \preceq t'$.

A recent approach uses an extension of simulation relations, called label-dominance simulation, in order to compute a dominance relation (Torralba and Hoffmann 2015). A key idea is that, if we compute a simulation relation on each LTS, $\preceq_1, \ldots, \preceq_k$, they can be combined to derive a dominance relation on the global LTS, $\preceq$ such that $(s_1, \ldots s_k) \preceq (t_1, \ldots t_k) iff s_1 \preceq t_1, \ldots, s_k \preceq_k$. This is a tractable dominance pruning approach, since the coarsest simulation relation can be computed in time polynomial in the size of the LTS (Henzinger, Henzinger, and Kopke 1995).

Label-dominance simulation is able to obtain coarser relations by considering the relation between labels. A label l' **dominates** l **in** Θ **given** $\preceq$ if for every transition $s \xrightarrow{l} s'$ there exists a transition $s \xrightarrow{l'} t'$ s.t. $s' \preceq t'$.

Definition 1 *Let $X = \{\Theta_1, \ldots, \Theta_k\}$ be a set of LTSs sharing the same labels. Denote the states of Θ_i by S_i. A set $\mathcal{R} = \{\preceq_1, \ldots, \preceq_k\}$ of binary relations $\preceq_i \subseteq S_i \times S_i$ is a **label-dominance simulation** for X if, whenever $s \preceq_i t$, $s \in S_i^G$ implies that $t \in S_i^G$, and for every transition $s \xrightarrow{l} s'$ in Θ_i, there exists a transition $t \xrightarrow{l'} t'$ in Θ_i such that $c(l') \leq c(l)$, $s' \preceq_i t'$, and, for all $j \neq i$, l' dominates l in Θ_j given $\preceq_j$. We call $\mathcal{R}$ the **coarsest** label-dominance simulation if, for every label-dominance simulation $\mathcal{R}' = \{\preceq'_1, \ldots, \preceq'_k\}$ for X, $\preceq'_i \subseteq \preceq_i$ for all i.*

Even though label dominance and each relation, $\preceq_i$, depend on each other, a coarsest label-dominance simulation always exists and can be computed in polynomial time in the size of the LTSs (Torralba and Hoffmann 2015). Moreover, coarser relations may be obtained by introducing a $noop$ operator of zero cost such that every state has a self-loop transition labeled with $noop$.

Transformation of Planning Tasks in M&S

Even though M&S has been traditionally understood as a method to generate admissible abstraction heuristics, it is not necessarily limited to those purposes. In fact, the ability of representing a planning task as the synchronized product of several LTSs makes M&S an excellent framework to manipulate the input planning task in different contexts.

Operations on the individual LTSs possibly modify the global LTS, but preserve certain properties. Merge operations do not affect the global LTS and exact label reduction preserves its structure. Removing unreachable and dead-end abstract states does not affect the part of the state space that is both reachable and solvable. Finally, shrinking strategies

guarantee that the result is an abstraction of the original LTS, so they obtain admissible heuristics. Bisimulation shrinking is globally h-preserving since it guarantees that the resulting global LTS is a bisimulation of the original problem.

In this paper, we are interested in transformations that simplify the planning task by removing irrelevant parts while preserving at least one optimal plan. We say that a transformation is *plan-preserving* if it removes states and transitions from the problem while preserving at least one optimal plan. A set of states and transitions is *irrelevant* for a planning task Π if removing them from Π does not change the optimal solution cost, $h^*(\mathcal{I})$, i. e., at least one of the optimal plans of Π is preserved. Irrelevance pruning generalizes pruning unreachable and dead-end states, which are irrelevant by definition. Note that the evaluation of whether individual states or transitions are relevant cannot be done independently, since in tasks with more than one optimal solution it may happen that each of them is irrelevant on their own but at least one of them is necessary.

In abstract state spaces, each abstract transition may correspond to many transitions in the global LTS. We say that an abstract transition is *irrelevant* if all the induced transitions in the global LTS are irrelevant. Next, we describe a sufficient criterion to identify some *irrelevant* transitions that can be pruned. Our method uses label-dominance simulation relations to identify irrelevant transitions that can be removed while preserving the optimal solution cost $h^*(s)$ from every state s in the global LTS. Combining our method with reachability analysis we can prune large parts of the state space.

Subsumed Transitions

Simulation relations may be used in order to remove *irrelevant* transitions that are not needed for an optimal plan from any given state. The main idea is that we can remove any abstract transition if there is another transition that is strictly better, i. e., one that leads to a better abstract state and the effects of its label are at least as good in other LTSs.

Definition 2 *A transition $s_i \xrightarrow{l} t_i \in \mathcal{T}_i$ is **subsumed** if and only if there exists another transition $s_i \xrightarrow{l'} t'_i \in \mathcal{T}_i$ such that $t_i \preceq t'_i$ and l' dominates l in all Θ_j for $j \neq i$. In that case, we say that transition $s \xrightarrow{l} t$ is subsumed by $s \xrightarrow{l'} t'$.*

Just like in the definition of label-dominance simulations, we may assume a self-loop $noop$ transition in every state. Thus, a transition $s \xrightarrow{l} t$ is subsumed by $s \xrightarrow{noop} s$ if $t \preceq s$ and l is dominated by $noop$ in every other LTS. Intuitively, removing a subsumed transition from the problem is admissible because, whenever it is applicable, the alternative transition is also applicable and leads to an at least as good state.

Theorem 1 *Let $\{\Theta_1, \ldots, \Theta_k\}$ be a set of LTSs sharing the same labels. Then, pruning from Θ_i a transition $s_i \xrightarrow{l} t_i$ that is subsumed by another, $s_i \xrightarrow{l'} t'_i$, is globally h-preserving.*

Proof: We show that a strong optimal plan is preserved for every state s in the global LTS, Θ, by induction on $h^*(s)$ and $h^{0*}(s)$. This trivially holds for goal states, where $h^*(s) = h^{0*}(s) = 0$. Consider an arbitrary state $s = (s_1, \ldots, s_k)$ with a strong optimal plan starting with a transition $(s_1, \ldots, s_k) \xrightarrow{l} (t_1, \ldots, t_k)$. Suppose by induction that every state in Θ, n, such that $h^*(n) < h^*(s) \vee (h^*(n) = h^*(s) \wedge h^{0*}(n) < h^{0*}(s))$ has a strong optimal plan without any transition induced by $s_i \xrightarrow{l} t_i$. Then, as either $h^*(s)$ or $h^{0*}(s)$ monotonically decrease after applying each action in the strong optimal plan, we only need to show that s has another strong optimal plan that does not start with any transition induced by $s_i \xrightarrow{l} t_i$. As l' dominates l for all Θ_j, $j \neq i$, necessarily $c(l') \leq c(l)$ and l' is applicable in s to reach $t' = (t'_0, \ldots, t'_k)$ such that $t_j \preceq t'_j$ for every j. By the definition of label-dominance simulation, $h^*(t') \leq h^*(t)$ and, if $h^*(t') = h^*(t)$ then $h^{0*}(t') \leq h^{0*}(t)$. Thus, there exists an alternative strong optimal plan for s that starts with $s \xrightarrow{l'} t'$ and does not contain any transition induced by $s_i \xrightarrow{l} t_i$. □

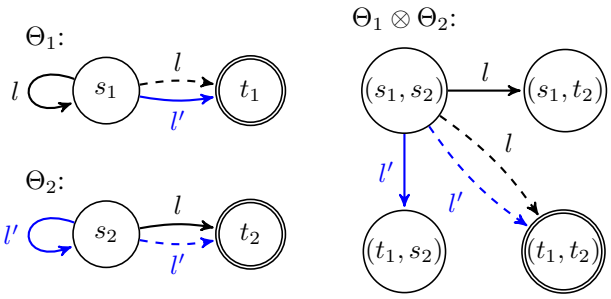

Figure 1: Example where pruning more than one transition at once is not safe. Dashed transitions in Θ_1 and Θ_2 are subsumed and induce the dashed transitions in the global LTS.

Note, however, that the above proof is only about pruning a **single** transition. In fact, it is **not safe** to prune more than one transition at once. One must be careful to avoid removing equivalent transitions when they subsume each other. In those cases, Definition 2 must be applied to remove one transition at a time, avoiding the elimination of all the alternatives. Moreover, after pruning each transition the label dominance might change, which in turn may require updating the simulation relations, as they might have changed as well.

To illustrate this, consider the example in Figure 1. In that example, label l dominates label l' in Θ_1, i. e., in all but Θ_2. Similarly, l' dominates l in all but Θ_1. That means that in Θ_1, we can prune the transition $s_1 \xrightarrow{l} t_1$, while in Θ_2 we can prune $s_2 \xrightarrow{l'} t_2$. However, after merging the two LTSs, resulting in $\Theta_1 \otimes \Theta_2$, displayed on the right hand side of Figure 1, there is no longer any connection from (s_1, s_2) to (t_1, t_2), while without pruning there are transitions $(s_1, s_2) \xrightarrow{l} (t_1, t_2)$ and $(s_1, s_2) \xrightarrow{l'} (t_1, t_2)$, so that $h^*((s_1, s_2))$ clearly has changed. The reason for this behavior is that after removing $s_1 \xrightarrow{l} t_1$ from Θ_1, the label dominance relation changes: label l no longer dominates l' in Θ_1 so that according to Definition 2 the transition $s_2 \xrightarrow{l'} t_2$ is no longer subsumed and cannot be pruned.

Updating the label-dominance simulation requires to propagate the changes to all simulation relations for every removed transition. This is not feasible in practice, so instead we decided to avoid pruning any transition if doing so will change the label-dominance relation. This saves all the recalculation overhead while still retaining safety, though it may decrease the amount of pruning achieved.

Pruning operators

One important application of irrelevant transition pruning is that it can be used to identify operators that can be removed from the planning task, preserving the optimal solution. A label l is dead if there is no transition labeled with l in some Θ_i (Sievers, Wehrle, and Helmert 2014). By the definition of synchronized product, dead labels do not induce any transition in the global LTS. Thus, removing dead labels from all the individual LTSs is an exact transformation that does not affect the global LTS. Sievers, Wehrle, and Helmert used this fact to prove that their label reduction is exact. Next, we show that if a label is dead, the corresponding operators can safely be removed from the planning task.

Theorem 2 *Let Π be a planning task with state space Θ. Let $\{\Theta_1, \ldots, \Theta_k\}$ be a set of LTSs such that $\bigotimes_i \Theta_i$ is a* plan-preserving transformation *of Θ. Let l be a dead label in any Θ_i. Then Π', which results from removing the set of operators associated with l, $\mathcal{O}_l$, has an optimal solution that is also valid for Π.*

Proof: As removing operators cannot possibly generate any new plans, any plan for Π' is a plan for Π. That at least one optimal plan is preserved in Π' follows from the definition of a *plan-preserving* transformation. As $\bigotimes_i \Theta_i$ is a *plan-preserving* transformation of Θ, it has an optimal plan for Π. This plan is preserved in Π' since l is dead in some Θ_i, $\mathcal{O}_l$ do not induce any transition in $\bigotimes_i \Theta_i$, so any plan for $\bigotimes_i \Theta_i$ is a plan for Π'. □

Even though previous M&S approaches could be used to remove operators from the planning task, these were limited to operators that are never applicable or that always lead to dead-end states. With the subsumed transition pruning presented in the previous section, operators that are not needed to obtain an optimal plan can be removed, even if they could be used in valid plans. In the logistics example with a single truck and several packages from the Background section, all the packages can reach all locations so that all operators are applicable. However, after subsumed transition pruning and unreachability analysis, all operators unloading a package in a non-goal position or loading a package in any location except the initial location can be detected as irrelevant.

Irrelevance Pruning for M&S Heuristics

Merge-and-shrink is commonly used to generate admissible abstraction heuristics for cost-optimal planning. To do so, M&S iterates merging and shrinking steps until a single LTS remains. The result is an abstraction of the original planning task, i. e., it induces an admissible heuristic. Pruning unreachable states might cause the heuristic to produce inadmissible estimates but only for unreachable states, so that the heuristic is still admissible for every state in the search.

Since pruning subsumed transitions is globally h-preserving, it also preserves admissibility of the M&S heuristic. However, this is not true in general for irrelevant transition pruning and, in particular, when combining subsumed transition and unreachable state pruning. The underlying reason is that subsumed transition pruning does not preserve initial state distances, so that new states can become unreachable. Since the heuristic is not guaranteed to be admissible for unreachable states, the final heuristic might return inadmissible estimates for states in the search.

As an extreme case, consider an M&S abstraction where shrinking has caused the initial state to be a goal. Then, irrelevance pruning will prune every other state, and the resulting heuristic is inadmissible and unsafe. Nevertheless, this does not prevent us from using irrelevance pruning to derive heuristics for cost-optimal planning. If irrelevance pruning is performed before using arbitrary shrinking strategies, the resulting heuristic can be shown to be globally admissible.

Definition 3 *(Karpas and Domshlak 2012) A heuristic h is called* ***globally admissible*** *if, for any planning task Π, if Π is solvable, then there exists an optimal plan π for Π such that, for any state s along π, $h(s) \leq h^*(s)$.*

Theorem 3 *Let Θ' be a* plan-preserving *transformation of a state space Θ and $\alpha(\Theta')$ an abstraction of Θ'. Then, $h_{\alpha(\Theta')}$ is a globally admissible heuristic for Θ.*

Proof Sketch: From the definition of *plan-preserving* transformation, Θ' has an optimal plan that is also optimal for Θ. $h_{\alpha(\Theta')}$ is an abstraction heuristic so it is admissible for every state in the plan. □

Karpas and Domshlak proved that if A* uses a globally admissible heuristic, the resulting plan is guaranteed to be optimal. Intuitively, even if the heuristic is inadmissible for our task, Π, it is admissible for a task obtained through a *plan-preserving* transformation. Thus, irrelevance pruning can be used to derive globally admissible heuristics if performed before any shrinking is applied. Next, we show that it can be combined with label reduction and bisimulation.

Label Reduction and Bisimulation

Bisimulation shrinking with label reduction has been a key factor for the success of M&S heuristics. Bisimulation can reduce the abstraction size while preserving the goal-distance information (Nissim, Hoffmann, and Helmert 2011). *Exact* label reduction further pushes the benefits of bisimulation by considering some labels equivalent but not inducing any new transition in the state space (Sievers, Wehrle, and Helmert 2014). Even though irrelevance pruning cannot be interleaved with arbitrary shrinking, it is safe to do so with exact label reduction and bisimulation.

Proposition 1 *Let Θ_b be the result of applying exact label reduction, τ, and bisimulation shrinking γ to a state space, Θ. Let Θ'_b be a plan-preserving transformation of Θ_b. Then, there exists a plan-preserving transformation of Θ, Θ', s.t. Θ'_b is a bisimulation of $\tau(\Theta')$.*

Proof Sketch: For the case of exact label reduction, it does not induce any new transition in the global LTS. Therefore, if a transition labeled with $\tau(l)$ is irrelevant after label reduction, all the transitions labeled with l were irrelevant.

For bisimulation, consider a transition $s_b \xrightarrow{l} t_b$ that is irrelevant in Θ_b. We show that the set of transitions $s \xrightarrow{l} t$ s.t. $\gamma(s) = s_b$ and $\gamma(t) = t_b$ is irrelevant in Θ. This is straightforward for transitions that do not belong to the optimal solution of Θ. If some $s \xrightarrow{l} t$ lies on an optimal plan for Θ, then $s_b \xrightarrow{l} t_b$ lies on an optimal plan for Θ_b because bisimulation is globally h-preserving. Since $s_b \xrightarrow{l} t_b$ is irrelevant, there exists an alternative plan in Θ_b that contains a transition $s_b \xrightarrow{l'} u_b$. Again, by the properties of bisimulation, all $s' \in \Theta$ necessarily have a transition $s \xrightarrow{l'} u$ s.t. $\gamma(u) = t_b$. Therefore $s \xrightarrow{l} t$ is irrelevant as well. □

However, applying label reduction may significantly reduce the achieved pruning. Figure 2 shows an example where two transitions, $s_1 \xrightarrow{l} t_1$ and $s_1 \xrightarrow{l'} t_1$, can be pruned. Label l is dominated by l_1 in Θ_2 and $t_1 \preceq u_1$, so $s_1 \xrightarrow{l} t_1$ is subsumed by $s_1 \xrightarrow{l_1} u_1$. For similar reasons $s_1 \xrightarrow{l_2} v_1$ subsumes $s_1 \xrightarrow{l'} t_1$. In the example of Figure 2, labels l and l' can be reduced to a single label, because they are equivalent in all Θ_i except for one, Θ_2. Thus, aggregating those labels into a new label, $\tau(l)$, does not induce any new transitions in the overall system. However, after label reduction no transition can be pruned in the example anymore. The two transitions $s_1 \xrightarrow{l} t_1$ and $s_1 \xrightarrow{l'} t_1$ that could be pruned before are now transformed into a single $s_1 \xrightarrow{\tau(l)} t_1$. While l was dominated by l_1 and l' was dominated by l_2 in Θ_2, the new label $\tau(l)$ is not dominated by any other label. Thus, even though the two aggregated transitions could be pruned, $s_1 \xrightarrow{\tau(l)} t_1$ is not detected as subsumed.

Opposite examples where a transition becomes subsumed only after label reduction can be constructed as well. Thus, label reduction might increase or decrease the pruning.

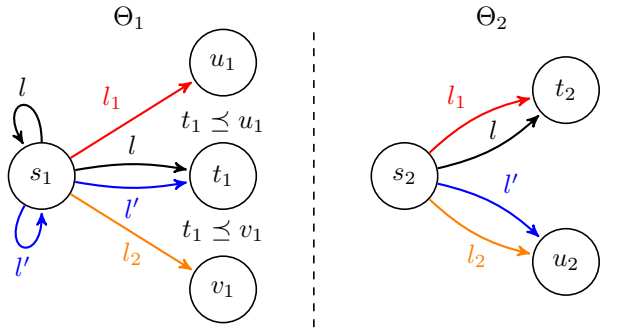

Figure 2: Example where label reduction decreases the amount of subsumed transitions.

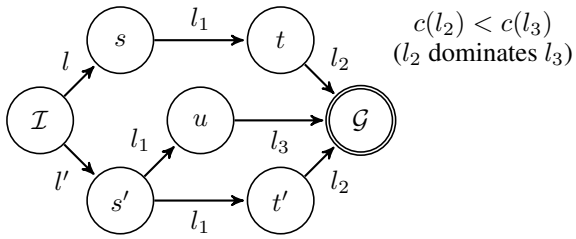

Figure 3: Example where similarity is coarser than bisimilarity (s and s' are similar but not bisimilar).

Another separate problem is that, after the labels of several operators are reduced to the same label, it will not become dead unless all the operators are dead, greatly decreasing the operator pruning power. To address this, whenever we use label reduction, we keep track of the operators that enabled a transition. For example, suppose that we have $s \xrightarrow{o_1} t$ and $s' \xrightarrow{o_2} t'$ and we reduce labels o_1 and o_2 to label l. Then, we have transitions $s \xrightarrow{l^{\{o_1\}}} t$ and $s' \xrightarrow{l^{\{o_2\}}} t'$. Whenever two transitions become equivalent (same source, label and target) because of label reduction or (bisimulation) shrinking, we compute the union of their sets of operators. Labels $l^{\{o_1\}}, l^{\{o_2\}}, and l^{\{o_1,o_2\}}$ are exactly the same label for all purposes, but we store the additional information to perform the operator pruning at the end of our process. If for some Θ_i, operator o does not belong to the set of operators of any transition, then o can be removed from the task.

Similarity Shrinking

In model-checking it is well-known that similarity equivalence implies bisimilarity but not vice versa (Henzinger, Henzinger, and Kopke 1995), so it is interesting to analyze the potential of simulations for shrinking strategies. Given a simulation relation, $\preceq$, two states s, s' are similar, $s \sim_s s'$, if and only if $s' \preceq s$ and $s \preceq s'$. Just like bisimulation shrinking aggregates bisimilar states according to the coarsest goal-respecting bisimulation, *similarity shrinking* consists of aggregating states that are similar according to a label-dominance simulation. The definition of a label-dominance simulation ensures that similarity shrinking is globally h-preserving, so that it derives the perfect heuristic values, like bisimulation.

Proposition 2 *Let $\{\Theta_1, \ldots, \Theta_k\}$ be a set of LTSs sharing the same labels and let $\preceq_1, \ldots, \preceq_k$ be a label-dominance simulation over $\{\Theta_1, \ldots, \Theta_k\}$. Then similarity shrinking of Θ_i according to $\preceq_i$ is globally h-preserving.*

Proof Sketch: $s_i \preceq s'_i$ implies that for any completion d, $h^*(s'_i \cup d) \leq h^*(s_i \cup d)$ and $s'_i \preceq s_i$ implies that $h^*(s_i \cup d) \leq h^*(s'_i \cup d)$. □

Similarity shrinking can reduce the size of the abstraction more than bisimulation as shown in the example of Figure 3. State t is bisimilar with t', since they have the same transition to $\mathcal{G}$ labeled with l_2. However, s and s' are not bisimilar because of $s' \xrightarrow{l_1} u$ ($u \not\sim_s t$, so $s \xrightarrow{l_1} t$ is not equivalent to

$s' \xrightarrow{l_1} u$). On the other hand, $u \preceq t \sim_s t'$ so s is similar with s' and they are aggregated by similarity shrinking. Bisimilarity implies similarity so similarity shrinking is at least as good as bisimulation but not vice versa.

Also, the label dominance relation can be used for exact label reduction. Whenever two labels l and l' dominate each other in all but one LTS, we say that they are *combinable*. In fact, any pair of combinable labels satisfies the criteria of exact label reduction after applying similarity shrinking in every Θ_j in which they are not equivalent.

The relation of bisimulation and similarity shrinking has a close relationship with subsumed transition pruning.

Proposition 3 *Let Θ be an LTS without any subsumed transition and without any pair of combinable labels. Then similarity and bisimulation shrinking of Θ are equivalent.*

Proof: Suppose the opposite. Then, there are two states s and s' in Θ such that they are similar, $s \sim_s s'$, but not bisimilar, $s \not\sim_b s'$. Since $s \not\sim_b s'$, one of them, say s, has a transition $s \xrightarrow{l} t$ such that there is no transition $s' \xrightarrow{l'} t'$ for any $t' \sim_b t$. But, as $s \preceq s'$, s' has a transition $s' \xrightarrow{l'} t'$ such that $t \preceq t'$ and l' dominates l in all other LTSs. Then, as $s' \preceq s$, s has a transition $s \xrightarrow{l''} t''$ such that $t' \preceq t''$ and l'' dominates l'. $l'' \neq l$ since otherwise l and l' are combinable. Since a simulation relation is transitive, $t \preceq t''$. Thus, $s \xrightarrow{l} t$ is subsumed by $s \xrightarrow{l''} t''$. □

Thus, the difference between similarity and bisimulation can be attributed to subsumed transitions and the simultaneous computation of label dominance and the simulation relations, whereas label reduction and bisimulation are independent processes. However, unlike bisimulation, it is not always safe to apply irrelevance pruning after similarity shrinking. For using the same proof as for Proposition 1, we require that all the labels l, l' such that they dominate each other in all Θ_j except one are reduced to the same label. Thus, we apply similarity shrinking and reduce all the labels that dominate each other in all but one LTS. However, this is not always possible. For example, suppose that there are labels l_1, l_2, and l_3 such that l_1 and l_2 dominate each other in all LTSs except Θ_1 and l_2 and l_3 dominate each other in all LTSs except Θ_2 but reducing the three labels, l_1, l_2, and l_3 to a single label is not exact. Thus, we only apply similarity shrinking whenever all combinable labels can be reduced.

Incremental Computation

In this paper we have devised two different applications for label-dominance simulations in M&S: pruning transitions and performing shrinking. Algorithm 1 shows the pseudocode of the M&S algorithm using label-dominance simulations, with the new parts highlighted in red.

In order to perform subsumed transition pruning and/or similarity shrinking at each step, the label-dominance simulation has to be recomputed at every iteration of the algorithm. As computing the label-dominance simulation is computationally expensive, this might become a bottleneck for the overall algorithm. Instead of recomputing it from scratch at every step of the algorithm, we apply an incremental computation of the simulation relation. To efficiently compute such an approximation of the label dominance relation in an incremental fashion, we retain the simulation relations of all previous LTSs unchanged, avoiding to recompute the coarser simulation except for the new LTS, a'. Also, instead of computing the simulation for new LTS a' from scratch, we check for which states s and t we already know that $s \preceq_{a'} t$. That is the case for all states $s = (s_1 \otimes s_2)$ and $t = (t_1 \otimes t_2)$ for which it holds that $s_1 \preceq_{a_1} t_1$ and $s_2 \preceq_{a_2} t_2$. We can skip checking these pairs, as we already know that the simulation relation holds; for all other pairs however we have to follow the same approach as before. By this incremental computation we lose the guarantee of having the coarsest label-dominance simulation, but the advantage in terms of runtime clearly outperforms this rather small loss in precision. At the end we recompute the coarsest label-dominance simulation from scratch, in order to increase the dominance pruning and find more irrelevant operators.

Algorithm 1: M&S + Label-Dominance Simulations

```
1  A ← AtomicAbstractions();
2  SIM ← ComputeLDSimulation(A);
3  while not done do
4      a1, a2 ← SelectMerge(A);
5      a' ← a1 ⊗ a2;
6      a' ← PruneUnreachableDeadEnd(a');
7      A, a' ← LabelReduction + Bisimulation(A, a') ;
8      SIM ← IncrementalLDSimulation(SIM, a');
9      a' ← PruneSubsumedTransitions(a', SIM) ;
10     a' ← ShrinkSimilarity(a', SIM);
11     A = (A \ {a1, a2}) ∪ {a'};
12 SIM ← ComputeLDSimulation(A);
13 A ← PruneSubsumedTransitions(A, SIM);
14 O' ← RemoveSubsumedOps(O, A);
15 return O', A, SIM;
```

The algorithm finishes whenever the abstraction sizes are larger than the established bounds. Then, it returns the set of relevant operators to be used in the search, as well as the current label-dominance simulation, and the set of current abstractions, which can be used to construct the dominance relation or an M&S heuristic, respectively.

Different variants of the algorithm are possible, as we explore in our experiments. For example, label reduction and exact shrinking may reduce the size of the abstractions, speeding up the algorithm and allowing more merges within the imposed size limit, but they can also (slightly) reduce the number of operators pruned. Also, if irrelevance pruning is only done at the end, the additional overhead of computing simulations at every step can be avoided.

Experiments

We perform two different experiments to measure the impact of subsumed transition pruning and similarity shrinking. First, we evaluate how they improve current M&S methods. Then, we use them for pruning irrelevant operators

and compare our approach against other irrelevance pruning techniques. Our techniques are implemented in Fast Downward (FD) (Helmert 2006). We ran all optimal-track STRIPS planning instances from the international planning competitions (IPC'98 – IPC'14). All experiments were conducted on a cluster of 2.20 GHz Intel Xeon E5-2660 machines, with time (memory) cut-offs of 30 minutes (4 GB).

We use the same configuration used in previous work for deriving simulation relations (Torralba and Hoffmann 2015). We run M&S with the DFP merge strategy (Sievers, Wehrle, and Helmert 2014) and a limit of 100,000 transitions and 300 seconds. As we have seen in this paper, the interaction of irrelevance pruning and label reduction or shrinking is not always positive. We tried different variations of Algorithm 1 disabling shrinking and/or the incremental computation of label-dominance simulations. We report the results of two configurations: P^i disables the shrinking of lines 7 and 10 in order to maximize the pruning achieved. P^b disables lines 8 and 9, avoiding the additional overhead of computing intermediate simulations. These configurations obtained the best results, though they are not far from other configurations.

M&S Heuristics

To evaluate the effect of subsumed transition pruning and similarity shrinking if used to derive admissible heuristics, we compare the baseline M&S heuristic against one initialized with the set of abstractions after irrelevance pruning/similarity shrinking. As shrinking strategy we use M&S with bisimulation and a limit of 100,000 states.

Similarity shrinking does not have any impact in most domains, except Parcprinter and Woodworking. As suggested by Proposition 3, it behaves as bisimulation except in domains with a huge number of subsumed transitions. Overall, the overhead of computing label-dominance simulations does not pay off and coverage slightly decreases with respect to the baseline. On the other hand, M&S with subsumed transition pruning, P^i, derives better heuristics increasing coverage by 8 problems. Using irrelevance pruning has mostly a positive impact on the number of expanded nodes and the planning time, as shown in Figure 4.

Operator Pruning Power

In this section, we evaluate the ability of M&S with subsumed transition pruning to eliminate operators from the planning task and how this improves the performance of

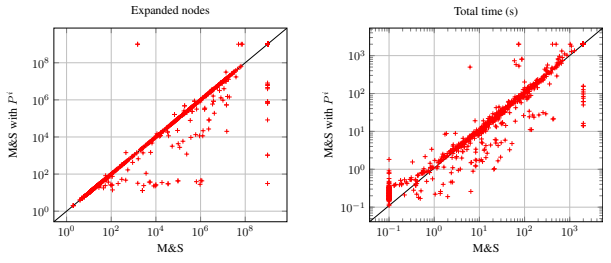

Figure 4: Subsumed transition pruning in M&S heuristics.

state-of-the-art cost-optimal planners. We run A* with LM-cut (Helmert and Domshlak 2009) with/without dominance pruning using the same simulation relations. In order to show how irrelevant operator pruning can be beneficial for other kind of planners, we run symbolic bidirectional brute-force search with the latest enhancements on symbolic search planning (Kissmann and Edelkamp 2011; Torralba, Edelkamp, and Kissmann 2013; Torralba and Alcázar 2013).

We also compare our results with the h^2-based preprocessor that prunes operators that are inconsistent with the state invariants (Alcázar and Torralba 2015). Since the pruning power of both techniques is complementary, we also combine them. In this case, we first apply the h^2 preprocessing stage and then run our simulation algorithm on the result.

Table 1 shows the percentage of pruned operators compared to the baseline (FD preprocessor), computed as the sum of pruned operators divided by the sum of total operators in all problems where the six preprocess configurations ended successfully. This gives more weight to larger instances, since they are less biased, for example, when pruning is only obtained if M&S constructs the entire state space and gets the perfect heuristic. We exclude domains with less than 5% pruning and unchanged coverage.

Overall, P^i obtains most pruning, with a geometric mean of 32% of the operators. However, the pruning power may vary a lot for different domains, pruning up to 90% of the operators (e. g., in Trucks). Unsurprisingly, more pruning is achieved in domains where dominance pruning worked best, such as logistics-like domains, Parcprinter or Woodworking. Our pruning is clearly orthogonal with h^2-based pruning, getting the best results when combining both techniques. The preprocess successfully finishes for most instances in the reach for optimal planners, except for a few exceptions.

Despite the time-consuming preprocessing stage, removing operators from the planning task is clearly effective to improve coverage across different planners. Even though coverage decreases in a few domains (mostly due to failures in the preprocess), total coverage is improved in all planners. Again, there is a synergy with h^2. The improvement of irrelevance pruning is reduced when the same simulation relations are used for dominance pruning, though it still improves coverage in several domains. Moreover, the method successfully improves symbolic planning as well, where it is not easy to implement dominance pruning efficiently.

Comparison Against Irrelevance Pruning Methods

A very related technique is path relevance analysis, which compares paths in the domain transition graphs (DTGs) that can be replaced by other paths. Replaceable paths are irrelevant and actions that only appear in such paths are irrelevant too. The connection with our approach is immediate, substituting path replaceability for subsumed transitions. The initial method proposed by Scholz (2004) has exponential complexity in the size of the planning task. Based on a previous adaptation of Scholz's method (Jonsson 2007), Haslum, Helmert, and Jonsson (2013) proposed a tractable approximation that we call HHJ. Since we are interested in tractable methods, we focus our comparison on the HHJ method.

Domain	#	Preprocess #P	Preprocess max time (s)	% Pruned operators – P^b	% Pruned operators – P^i	% Pruned operators h^2 P^b	% Pruned operators h^2 P^i	A* + LM-Cut –	A* + LM-Cut – P^b	A* + LM-Cut – P^i	A* + LM-Cut h^2 P^b	A* + LM-Cut h^2 P^i	A* + LM-Cut + Dominance pruning –	A* + LM-Cut + Dominance pruning – P^b	A* + LM-Cut + Dominance pruning – P^i	A* + LM-Cut + Dominance pruning h^2 P^b	A* + LM-Cut + Dominance pruning h^2 P^i	Sym-Bidir h^2	Sym-Bidir h^2 P^b	Sym-Bidir h^2 P^i
Airport	50	29	1473.94	10	13	**76**	**76**	**76**	28	−1	**0**	**+1**	**+1**	**0**	27	**0**	+1	**+2**	+1	−1	25	**−2**	**−2**
Barman11	20	20	384.19	1	1	13	**14**	**14**	4	**0**	**0**	**0**	**0**	**0**	4	**0**	**0**	**0**	**0**	**0**	10	**−1**	**−1**
Barman14	14	14	371.66	1	1	12	13	**14**	0	**0**	**0**	**0**	**0**	**0**	0	**0**	**0**	**0**	**0**	**0**	6	**0**	**0**
Child14	20	20	420.13	0	**10**	0	0	**10**	0	**0**	**0**	**0**	**0**	**0**	0	**0**	**0**	**0**	**0**	**0**	4	**0**	**0**
Depots	22	22	491.95	3	2	**21**	**21**	**21**	7	**0**	**0**	**0**	**0**	**0**	7	**0**	**0**	**0**	**0**	**0**	7	**0**	**0**
Floortile11	20	20	199.00	27	28	**38**	**38**	**38**	7	+1	+1	**+7**	**+7**	**+7**	8	**0**	**0**	**+6**	**+6**	**+6**	14	**0**	**0**
Floortile14	20	19	82.77	27	28	**38**	**38**	**38**	6	+2	+2	**+11**	**+11**	**+11**	8	**0**	**0**	**+9**	**+9**	**+9**	20	**0**	**0**
Gripper	20	20	187.73	**28**	11	0	**28**	11	7	**0**	**+1**	**0**	**0**	**+1**	**14**	**0**	−6	**0**	**0**	−6	20	**0**	**0**
Hiking14	20	20	127.44	0	**37**	0	0	**37**	9	**0**	**0**	**0**	**0**	**0**	9	**0**	**+2**	**0**	**0**	**+2**	15	**0**	**0**
Logistics00	28	27	234.14	61	**67**	0	60	**67**	20	**+1**	**+1**	**0**	**+1**	**+1**	20	**+1**	**+1**	**0**	**+1**	**+1**	18	**+4**	**+4**
Logistics98	35	29	1698.06	37	**42**	0	37	**42**	6	**0**	**0**	**0**	**0**	**0**	6	**0**	**0**	**0**	**0**	**0**	5	**+1**	**+1**
Maintenance	5	5	39.35	43	**76**	16	54	**76**	5	**0**	**0**	**0**	**0**	**0**	5	**0**	**0**	**0**	**0**	**0**	5	**0**	**0**
Miconic	150	150	1249.61	34	**58**	0	34	**58**	141	**+1**	**+1**	**0**	**+1**	**+1**	141	**+1**	**+1**	**0**	**+1**	**+1**	114	**−1**	−3
Mprime	35	28	814.39	0	0	**10**	**10**	**10**	22	**0**	**0**	**+1**	**+1**	**0**	22	**0**	**0**	**+1**	**+1**	**0**	23	**0**	**0**
Mystery	30	17	421.83	0	0	**30**	**30**	**30**	**17**	**0**	−1	**0**	**0**	**0**	**17**	**0**	−1	**0**	**0**	**0**	15	**0**	**0**
NoMystery	20	20	498.77	13	**49**	23	29	**49**	14	**+4**	**+4**	**0**	**+4**	**+4**	**20**	**0**	−1	**0**	**0**	**0**	14	**+6**	**+6**
OpenStack08	30	30	452.35	0	0	0	0	0	21	**0**	**0**	**0**	**0**	**0**	21	**0**	**0**	**0**	**0**	**+1**	30	**0**	**0**
OpenStack11	20	20	467.34	0	0	0	0	0	16	**0**	**0**	**0**	**0**	**0**	16	**0**	**0**	**0**	**0**	**+1**	20	**0**	**0**
ParcPrint08	30	28	377.22	73	78	71	**80**	**80**	18	**+6**	**+6**	+4	**+6**	**+6**	18	+5	+4	+4	+5	**+6**	21	**+1**	**0**
ParcPrint11	20	20	372.92	72	77	70	78	**79**	13	**+6**	**+6**	+4	**+6**	**+6**	13	+5	+4	+4	+5	**+6**	16	**+1**	**0**
Path-noneg	30	30	648.41	26	**38**	0	26	**38**	5	**0**	**0**	**0**	**0**	**0**	5	**0**	**0**	**0**	**0**	**0**	5	**+1**	**+1**
PegSol08	30	30	628.93	0	0	17	18	**19**	**28**	**0**	−1	**0**	**0**	**0**	**28**	**0**	−1	−1	−1	−1	29	**0**	**0**
PegSol11	20	20	204.54	0	0	**5**	**5**	**5**	**18**	**0**	−1	**0**	**0**	−1	**18**	**0**	**0**	−1	−1	−1	19	**0**	**0**
PipesNoTank	50	44	1489.25	0	0	**6**	**6**	**6**	17	**0**	**0**	**0**	**0**	**0**	17	**0**	**0**	**0**	**0**	**0**	16	**0**	**0**
PipesTank	50	37	728.48	0	0	**2**	**2**	**2**	**12**	−1	−1	**0**	−1	−1	12	**0**	**0**	**0**	**0**	**0**	17	**0**	**0**
Psr-small	50	50	1277.31	2	**85**	0	2	**85**	49	**0**	**0**	**0**	**0**	**0**	49	**0**	**0**	**0**	**0**	**0**	50	**0**	**0**
Rovers	40	35	1072.91	57	**71**	0	59	**71**	7	+2	**+3**	**0**	+2	+2	9	**0**	**+1**	**0**	**0**	**+1**	14	**+1**	**+1**
Satellite	36	23	1243.97	34	**50**	0	37	**50**	7	+1	**+2**	**0**	+1	**+2**	10	**+1**	−1	−1	−1	−1	10	**0**	**0**
Scanalyzer08	30	23	662.20	1	1	**16**	**16**	**16**	**15**	−2	−3	**0**	−2	−2	**14**	−1	−3	**0**	−1	−1	12	**0**	**0**
Scanalyzer11	20	15	687.03	0	0	**23**	**23**	**23**	**12**	−2	−3	**0**	−2	−2	**11**	−1	−2	**0**	−1	−1	9	**0**	**0**
Sokoban08	30	30	782.38	7	8	**23**	**23**	**23**	29	**0**	**0**	**0**	**0**	**0**	28	**+1**	**+1**	**+1**	**+1**	**+1**	26	**0**	**0**
Sokoban11	20	20	830.79	8	9	**27**	**27**	**27**	20	**0**	**0**	**0**	**0**	**0**	20	**0**	**0**	**0**	**0**	**0**	20	**0**	**0**
Tetris	17	11	1579.70	0	18	**91**	**91**	**91**	6	−2	−2	**+3**	+2	**+3**	4	**0**	+1	**+4**	+3	**+4**	9	**0**	**0**
Tidybot11	20	13	549.50	0	0	**73**	**73**	**73**	14	−1	−1	**+3**	+2	**+3**	12	**0**	−1	**+5**	**+5**	**+5**	**16**	**0**	−1
Tidybot14	20	3	568.73	0	0	**48**	**48**	**48**	9	−6	−6	**+4**	+3	+3	2	**0**	**0**	+7	+7	**+9**	11	**−2**	**−2**
TPP	30	25	919.25	43	25	56	60	**61**	6	**0**	**+1**	**0**	**0**	**+1**	7	**0**	**0**	**0**	**0**	**0**	8	**0**	**0**
Transport08	30	30	541.17	0	0	0	0	0	11	**0**	**0**	**0**	**0**	**0**	11	**0**	**0**	**0**	**0**	**0**	**14**	**0**	−1
Transport14	20	20	354.97	0	0	0	0	0	6	**0**	**0**	**0**	**0**	**0**	6	**0**	**0**	**0**	**0**	**0**	8	**+1**	**+1**
Trucks	30	22	812.52	65	**90**	38	67	**90**	10	**0**	**+1**	**0**	**0**	**+1**	10	**0**	**+1**	**0**	**0**	**+1**	12	**0**	**+1**
VisitAll11	20	20	1723.16	3	**6**	0	3	**6**	10	**0**	**0**	**0**	**0**	**0**	11	**0**	**0**	**0**	**0**	**0**	12	**0**	**0**
Woodwork08	30	30	486.19	65	85	50	74	**88**	17	+8	**+11**	+5	+10	**+11**	24	+3	+4	+4	**+5**	**+5**	27	**+2**	**+2**
Woodwork11	20	20	467.72	63	**89**	51	73	88	12	+7	**+8**	+3	+7	**+8**	17	+2	**+3**	**+3**	**+3**	**+3**	19	**+1**	**+1**
Zenotravel	20	18	901.85	42	**43**	0	42	**43**	13	**0**	**0**	**0**	**0**	**0**	13	**0**	**0**	**0**	**0**	**0**	10	**0**	**0**
Others	340	336		< 5	< 5	< 5	< 5	< 5	139	0	0	0	0	0	139	0	0	0	0	0	189	0	0
Total	1612	1463		20	32	23	33	**42**	833	+24	+29	+46	+60	**+65**	853	+17	+8	+47	+48	**+50**	964	**+13**	+8

Table 1: Operator pruning with M&S and simulation. Number of tasks completed by all preprocess (#P) and maximum preprocess time. Percentage of pruned operators over instances successfully preprocessed by all configurations and relative coverage. There are six different sets of operators, depending on whether h^2 is used and the type of irrelevance pruning, P^i, P^b, or none.

Domain	Operators P^i	Operators HHJ	LM-Cut –	LM-Cut P^i	LM-Cut HHJ
Blocksworld	0.01	**0.81**	28	28	**35**
Driverlog	**0.05**	**0.05**	13	13	**14**
Logistics00	**0.65**	0.52	20	**21**	**21**
Logistics98	**0.38**	0.09	6	6	6
Miconic	**0.58**	0.57	141	**142**	**142**
Total			208	210	**218**

Table 2: Comparison against path-relevance analysis.

Relevance-path analysis is more general in the sense that it considers path instead of transition subsumption. However, even if paths are not explicitly considered, simulation relations implicitly consider paths to the goal. On the other hand, our technique works in an arbitrary set of M&S abstractions, instead of being formulated in terms of the DTGs of the problem variables, which is similar to being restricted to atomic abstractions. The main limitation of the HHJ method is that it is specialized for unary domains and, although a generalization to non-unary domains is possible, it is a challenging task. Their path subsumption is based on comparing the preconditions of the operators in the path, but not the effects. This is solved by our label dominance relation, which is able to compare labels in non-unary domains.

Table 2 compares both techniques in the four unary domains that are supported by HHJ: Miconic, Logistics, Driverlog and Blocksworld. Even though unary domains might be favorable for HHJ, P^b has similar performance in terms of operator pruning and coverage, except in Blocksworld. However, it is worth noticing that HHJ uses a different (unary) formulation of Blocksworld.

Conclusions

Merge-and-shrink, commonly used to derive admissible abstraction heuristics, can also be used to transform planning tasks in different contexts. In this paper, we tailored M&S to remove irrelevant operators from the task prior to the search.

We presented subsumed transition pruning, a new operation for the M&S framework that uses label-dominance simulations to prune irrelevant transitions from abstract state spaces. We proved that irrelevance pruning can be combined with other M&S operations such as *exact* label reduction

and bisimulation to derive globally admissible heuristics for cost-optimal planning. This allows M&S to derive better heuristics and makes it a powerful tool for relevance analysis. We also presented similarity shrinking, a new shrinking strategy to derive perfect heuristics. We show that even though it dominates bisimulation, it is redundant with subsumed transition pruning and in general it does not payoff.

Despite the good results, the potential of simulation-based irrelevance pruning has not been fully explored yet. Extending the algorithm to reason with subsumed paths or allowing a transition to be subsumed by a set of transitions instead of by a single one are promising directions for future research.

Acknowledgments. We thank the anonymous reviewers, whose comments helped to improve the paper. This work was partially supported by the EU FP7 Programme under grant agreement 295261 (MEALS).

References

Alcázar, V., and Torralba, Á. 2015. A reminder about the importance of computing and exploiting invariants in planning. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*.

Dräger, K.; Finkbeiner, B.; and Podelski, A. 2006. Directed model checking with distance-preserving abstractions. In *Proceedings of the 13th International SPIN Workshop (SPIN 2006)*, volume 3925 of *Lecture Notes in Computer Science*, 19–34.

Dräger, K.; Finkbeiner, B.; and Podelski, A. 2009. Directed model checking with distance-preserving abstractions. *STTT* 11(1):27–37.

Grumberg, O., and Long, D. E. 1994. Model checking and modular verification. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 16(3):843–871.

Hall, D.; Cohen, A.; Burkett, D.; and Klein, D. 2013. Faster optimal planning with partial-order pruning. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*.

Haslum, P.; Helmert, M.; and Jonsson, A. 2013. Safe, strong, and tractable relevance analysis for planning. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*.

Haslum, P. 2007. Reducing accidental complexity in planning problems. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, 1898–1903.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, 162–169.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge & shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the Association for Computing Machinery* 61(3).

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS'07)*, 176–183.

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

Henzinger, M. R.; Henzinger, T. A.; and Kopke, P. W. 1995. Computing simulations on finite and infinite graphs. In *36th Annual Symposium on Foundations of Computer Science.*, 453–462.

Jonsson, A. 2007. Efficient pruning of operators in planning domains. In *Current Topics in Artificial Intelligence, 12th Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2007, Salamanca, Spain, November 12-16, 2007. Selected Papers*, volume 4788 of *Lecture Notes in Computer Science*, 130–139.

Karpas, E., and Domshlak, C. 2012. Optimal search with inadmissible heuristics. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*.

Kissmann, P., and Edelkamp, S. 2011. Improving cost-optimal domain-independent symbolic planning. In *Proceedings of the 25th National Conference of the American Association for Artificial Intelligence (AAAI-11)*, 992–997.

Kissmann, P. 2012. *Symbolic Search in Planning and General Game Playing*. Ph.D. Dissertation, Universität Bremen.

Loiseaux, C.; Graf, S.; Sifakis, J.; Bouajjani, A.; and Bensalem, S. 1995. Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design* 6(1):11–44.

McMillan, K. L. 1993. *Symbolic Model Checking*. Kluwer Academic Publishers.

Milner, R. 1971. An algebraic definition of simulation between programs. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence (IJCAI-71)*, 481–489.

Nissim, R.; Hoffmann, J.; and Helmert, M. 2011. Computing perfect heuristics in polynomial time: On bisimulation and merge-and-shrink abstraction in optimal planning. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*, 1983–1990.

Scholz, U. 2004. *Reducing Planning Problems by Path Reduction*. Ph.D. Dissertation, Technische Universität Darmstadt.

Sievers, S.; Wehrle, M.; and Helmert, M. 2014. Generalized label reduction for merge-and-shrink heuristics. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14)*, 2358–2366.

Torralba, Á., and Alcázar, V. 2013. Constrained symbolic search: On mutexes, BDD minimization and more. In *Proceedings of the 6th Annual Symposium on Combinatorial Search (SOCS'13)*, 175–183.

Torralba, Á., and Hoffmann, J. 2015. Simulation-based admissible dominance pruning. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*.

Torralba, Á.; Edelkamp, S.; and Kissmann, P. 2013. Transition trees for cost-optimal symbolic planning. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, 206–214.