Simulation-Based Admissible Dominance Pruning

Álvaro Torralba and Jörg Hoffmann
Saarland University
Saarbrücken, Germany
torralba@cs.uni-saarland.de, hoffmann@cs.uni-saarland.de

Abstract

In optimal planning as heuristic search, admissible pruning techniques are paramount. One idea is dominance pruning, identifying states “better” than other states. Prior approaches are limited to simple dominance notions, like “more STRIPS facts true” or “higher resource supply”. We apply simulation, well-known in model checking, to compute much more general dominance relations based on comparing transition behavior across states. We do so effectively by expressing state-space simulations through the composition of simulations on orthogonal projections. We show how simulation can be made more powerful by intertwining it with a notion of label dominance. Our experiments show substantial improvements across several IPC benchmark domains.

1 Introduction

Heuristic search is the predominant approach to cost-optimal planning. But the number of states that must be explored to prove optimality often grows exponentially even when using extremely well-informed heuristics [Helmert and Röger, 2008]. Therefore, recent years have seen substantial effort devoted to identifying and exploiting structure allowing to prune redundant parts of the state space. Known techniques of this kind pertain to symmetries (e. g. [Fox and Long, 1999; 2002; Domshlak et al., 2012]), partial-order reduction based methods like expansion-core [Chen and Yao, 2009] or strong stubborn sets [Valmari, 1989; Wehrle and Helmert, 2012; Wehrle et al., 2013; Wehrle and Helmert, 2014], and dominance pruning [Hall et al., 2013]. We follow up on the latter here.

Dominance pruning is based on identifying states “better than” other states. For example, a Logistics task where one truck must carry several packages to location G. Consider the position of any one package p. 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. We refer to this kind of relation between states as a dominance relation. (Hall et al. use the term “partial-order”, which we change here to avoid ambiguity with, e. g., partial-order reduction.)

Two main questions need to be answered: (1) How to discover the dominance relation? (2) How to simplify the search (and/or planning task) given a dominance relation? Hall et al. answered (2) in terms of an admissible pruning method, pruning state s if a dominating state t, with an at-most-as-costly path, has already been seen. We follow that idea here, contributing a BDD implementation. Our main contribution regards (1). Hall et al. use dominance relations characterized by consumed resources: state t dominates state s if s and t are identical except that t(r) ≥ s(r) for all resources r.1 Herein, we instead find the dominance relation through simulation, used in model checking mainly to compare different system models [Milner, 1971; Gentilini et al., 2003].

A simulation is a relation ≤ on states where, whenever s ≤ t, for every transition s → s′ there exists a transition t → t′ using the same action, such that s′ ≤ t′. In words, t simulates s if anything we can do in s, we can do also in t, leading to a simulating state. (For the reader familiar with the use of bisimulation in merge-and-shrink [Helmert et al., 2014]: simulation is “one half of” bisimulation.) A simulation clearly qualifies as a dominance relation. But how to find a simulation on the state space?

We employ a compositional approach, obtaining our simulation relation on the state space from simulation relations on orthogonal projections, i. e., projections whose variable subsets do not overlap. We enhance simulation with a concept of label (action) dominance, in addition to states. In our Logistics example above, e. g., for each package this detects the described relation (G is better than being in the truck is better than being at any location other than G). This yields a very strong dominance relation that allows to ignore any state in which a package is unnecessarily unloaded at an irrelevant location. Empirically, we find that indeed our pruning method often substantially reduces the number of expanded nodes.

For space reasons, we omit some proofs. Full proofs, and more examples, will be made available in a TR.

1Precisely, Hall et al. consider numeric state variables r and analyze whether higher r is always good, or is always bad, or neither. In Metric-FF’s [Hoffmann, 2003] linear normal form, this is equivalent to the formulation above. Hall et al. also handle STRIPS facts, as variables with domain {0, 1}. But, there, their notions trivialize to “t dominates s if t ⊇ s”.

1689
2 Background

A planning task is a 4-tuple \( (V, A, I, G) \). \( V \) is a finite set of variables \( v \), each \( v \in V \) being associated with a finite domain \( D_v \). A partial state over \( V \) is a function \( s \) on a subset \( V(s) \) of \( V \), so that \( s(v) \in D_v \), for all \( v \in V(s) \); \( s \) is a state if \( V(s) = V \). The initial state \( I \) is a state. The goal \( G \) is a partial state. A is a finite set of actions, each \( a \in A \) being a pair \((\text{pre}_a, \text{eff}_a)\) of partial states, called its precondition and effect. Each \( a \in A \) is also associated with its non-negative cost \( c(a) \in \mathbb{R}^+_0 \).

A labeled transition system (LTS) is a tuple \( \Theta = (S, L, T, s_0, S_G) \) where \( S \) is a finite set of states, \( L \) is a finite set of labels each associated with a label cost \( c(l) \in \mathbb{R}^+_0 \), \( T \subseteq S \times L \times S \) is a set of transitions, \( s_0 \in S \) is the start state, and \( S_G \subseteq S \) is the set of goal states.

The state space of a planning task is the LTS \( \Theta_H \) where: \( S \) is the set of all states; \( s_0 \) is the initial state I of \( \Pi \); \( s \in S_G \) iff \( G \subseteq s \); the labels \( L \) are the actions \( A \), and \( s \xrightarrow{a} s' \) is a transition in \( T \) if \( s \) complies with \( \text{pre}_a \), and \( s'(v) = \text{eff}_a(v) \) for \( v \in V \setminus \text{eff}_a(v) \) while \( s'(v) = s(v) \) for \( v \in V \setminus \text{eff}_a(v) \). A plan for a state \( s \) is a path from \( s \) to any \( s_G \in S_G \). The cost of the cheapest plan for \( s \) is denoted \( h^*(s) \). A plan for \( s_0 \) is a plan for \( \Pi \), and is optimal iff its cost equals \( h^*(s_0) \). As defined by Wehrle and Helmert [2014], a plan for \( s \) is strongly optimal if its number of 0-cost actions is minimal among all optimal plans for \( s \).

Abstractions and abstract state spaces are quite common in planning (e.g. [Helmert et al., 2007]). We build on this work, but only indirectly. We use merge-and-shrink abstractions as the basis from which our simulation process starts. That process itself will be described in a generic form not relying on these specific constructs. Hence, in what follows, we provide only a summary view that suffices to present our contribution.

Say we have a task \( \Pi = (V, A, I, G) \) with state space \( \Theta_H = (S, A, I, G, S_G) \), and a variable subset \( W \subseteq V \). The projection onto \( W \) is the function \( \pi^W : S \rightarrow S^W \) from the states \( S \) over \( V \) into the states \( S^W \) over \( W \), where \( \pi^W(s) \) is the restriction of \( s \) to \( W \). The projected state space \( \Theta^W_H \) is the LTS \((S^W, A, T^W, \pi^W(I), S^W_0)\), where \( T^W := \{ (\pi^W(s), l, \pi^W(s')) \mid (s, l, s') \in T \} \) and \( S^W_0 := \{ \pi^W(s_G) \mid s_G \in S_G \} \). Given two variable subsets \( W \) and \( U \), the (projections onto) \( W \) and \( U \) are orthogonal if \( W \cap U = \emptyset \). Orthogonality ensures the following reconstruction property: \( \Theta^W_H \otimes \Theta^U_H = \Theta^{W \cup U}_H \) for orthogonal \( W \) and \( U \), where \( \otimes \) is the synchronized product operation. Namely, given any two labeled transition systems \( \Theta^1 = (S^1, L, T^1, s^1_0, S^1_G) \) and \( \Theta^2 = (S^2, L, T^2, s^2_0, S^2_G) \) that share the same set \( L \) of labels, \( \Theta^1 \otimes \Theta^2 \) is the labeled transition system with states \( S^1 \times S^2 \), labels \( L \), transition \((s_1, s_2) \xrightarrow{l} (s'_1, s'_2) \) iff \( s_1 \xrightarrow{l} s'_1 \) in \( T^1 \) and \( s_2 \xrightarrow{l} s'_2 \) in \( T^2 \), start state \((s_0^1, s_0^2)\), and goal states \(((S^1_G, s_0^2), s_0^2) \cup (s_0^1, S^2_G) \cup (s_0^1, s_0^2) \cup (s_0^1, s_0^2))\).

Merge-and-shrink abstractions [Helmert et al., 2007; 2014] construct more general abstraction functions, and the corresponding abstract state spaces, by starting from atomic abstractions (projections onto single state variables), and inter-leaving merging steps (replacing two abstractions with their synchronized product) with shrinking steps (replacing an abstraction with an abstraction of itself). It has been shown that, if every shrinking step replaces the selected abstraction with a bisimulation of itself, then the final abstraction is a bisimulation of the overall state space \( \Theta_H \). It has also been shown that such shrinking can be combined with exact label reduction [Sievers et al., 2014]. A label reduction is a function \( \tau \) from the labels \( L \) into a set \( L^\tau \) of reduced labels preserving label cost i.e. \( c(l) = c(\tau(l)) \). Given an LTS \( \Theta \), denote by \( \tau(\Theta) \) the LTS identical to \( \Theta \) except that all labels have been replaced by \( \tau(l) \). Given a set \( \{\Theta^1, \ldots, \Theta^k\} \) of LTSs sharing labels \( L \), a label reduction \( \tau \) is exact if \( \tau(\Theta^1) \otimes \cdots \otimes \tau(\Theta^k) = \tau(\Theta^1 \otimes \cdots \otimes \Theta^k) \). Reducing labels in this way, and using bisimulation shrinking, merge-and-shrink delivers a bisimulation of \( \tau(\Theta_H) \).

3 Simulation Relations

Given a planning task with states \( S \), a dominance relation is a binary relation \( \preceq \subseteq S \times S \) where \( s \preceq t \) implies \( h^*(t) \leq h^*(s) \) and, if \( h^*(t) = h^*(s) \) then \( h^o(t) \leq h^o(s) \). This is exactly what is needed for admissible pruning during search, as discussed in the next section.

To find dominance relations in practice, we focus on the special case of simulation relations. These are well known in model-checking (e.g. [Grimberg and Long, 1994; Loiseaux et al., 1995]). Here we use a variant adapted to planning (only) in making explicit the distinction between goal states and non-goal states:

**Definition 1 (Simulation)** Let \( \Theta = (S, L, T, s_0, S_G) \) be an LTS. A binary relation \( \preceq \subseteq S \times S \) is a simulation for \( \Theta \) if, whenever \( s \leq t \) (in words: \( t \) simulates \( s \)), for every transition \( s \xrightarrow{l} s' \) there exists a transition \( t \xrightarrow{l'} t' \) s.t. \( s' \preceq t' \). We call \( \preceq \) goal-respecting for \( \Theta \) if, whenever \( s \leq t \), \( s \in S_G \) implies that \( t \in S_G \).

We call \( \preceq \) the coarsest goal-respecting simulation if, for every goal-respecting simulation \( \preceq' \), we have \( \preceq' \subseteq \preceq \).

A unique coarsest goal-respecting simulation always exists and can be computed in time polynomial in the size of \( \Theta \) [Henzinger et al., 1995]. Note that the coarsest simulation is always reflexive, i.e., \( s \preceq s \); the same is true of all simulation relations considered here. Intuitively, every state “dominates itself”.

Observe that \( s \preceq t \) implies \( h^*(t) \leq h^*(s) \), because any plan for \( s \) can be also applied to \( t \). Hence a goal-respecting simulation over states is a dominance relation. But, for obtaining that property, goal-respecting simulation is unnecessarily strict. It suffices to preserve, not the label \( l \) of the transition \( s \xrightarrow{l} s' \), but only its cost:

**Definition 2 (Cost-Simulation)** Let \( \Theta = (S, L, T, s_0, S_G) \) be an LTS. A binary relation \( \preceq \subseteq S \times S \) is a cost-simulation for \( \Theta \) if, whenever \( s \preceq t \), \( s \in S_G \) implies that \( t \in S_G \), and for every transition \( s \xrightarrow{l} s' \) there exists a transition \( t \xrightarrow{l'} t' \) s.t. \( s' \preceq t' \) and \( c(l') \leq c(l) \).

Hall et al. [2013] include an equivalent definition, calling it “compatibility” and not relating it to simulation.
A cost-simulation still is a dominance relation, and any goal-respecting simulation is a cost-simulation but not vice versa. However, in our compositional approach where individual dominance relations are computed on orthogonal projections, we need to preserve labels for synchronization across these projections. Hence, to ensure we obtain a dominance relation of the state space, we must use goal-respecting simulation (rather than cost-simulation) on each projection.

For our notion of label dominance, it will be important to consider LTSs with NOOPs added. For any LTS $\Theta$, we denote by $\Theta_{\text{noop}}$ the same LTS but with a new additional label $\text{noop}$ where $c(\text{noop}) = 0$ and, for every state $s$, a new transition $s \xrightarrow{\text{noop}} s$. Obviously, any dominance relation over $\Theta_{\text{noop}}$ is a dominance relation over $\Theta$. So for our purposes it suffices to find a dominance relation over the state space $(\Theta_{\Pi})_{\text{noop}}$ with NOOPs added.

4 Admissible Dominance Pruning

By $\text{A}^\ast$ with dominance pruning, we refer to the following modification of $\text{A}^\ast$: Whenever a node $N$ with state $s(N)$ and path cost $g(N)$ is generated, check whether there exists a node $N'$ in the open or closed lists, with state $s(N')$ and path cost $g(N')$, so that $s(N) \preceq s(N')$ and $g(N') \leq g(N)$. If so, prune $N$, i.e., do not insert it into the open list, nor into the closed list.

As $s \preceq t$ implies $h^*(t) \leq h^*(s)$, any plan through $N$ costs at least as much as an optimal plan through $N'$, so $\text{A}^\ast$ with dominance pruning guarantees optimality. In presence of 0-cost actions, one must be careful to not prune $s$ if this eliminates all possible plans for $t$. However, $s$ cannot belong to the strongly optimal plan of $t$ because $s \preceq t$ and $h^*(t) = h^*(s)$ implies $h^0(t) \leq h^0(s)$.

Dominance pruning can reduce $\text{A}^\ast$’s search space, but comes with a computational overhead. First, checking cost, the runtime required for checking whether there exists a node $N'$ in the open or closed lists, with the mentioned properties. Second, maintenance cost, the runtime and memory required for maintaining whichever data structure is used to keep checking cost (which is excessive in a naïve implementation) at bay. Depending on which of these two costs tend to be higher, variants of dominance pruning make sense.

Hall et al.’s [2013] dominance relations are characterized by resources $r$. They maintain, for each $r$, the set $S(r)$ of seen states with a positive value for $r$. Given a new state $s$, their check iterates over the states in the intersection of $S(r)$ for those $r$ where $s(r)$ is positive. This implementation has high checking cost but low maintenance cost. Hence Hall et al. perform the check not at node-generation time, but at node-expansion time, reducing the number of checks that will be made.

To deal with our much more general dominance relations, we developed a BDD-based [Bryant, 1986] implementation. This has low checking cost but high maintenance cost. Hence we perform the check at node-generation time, but only against the closed list, reducing the number of maintenance operations needed. We maintain a BDD $B_g$ for the set of states simulated by any $s(N')$ where $N'$ is a previously expanded node with $g(N') = g$. This is done for every $g$-value of the expanded nodes so far. Every time a node $N'$ is expanded, we determine the set of states $S_{\leq g(N')}$ simulated by $s(N')$, and add $S_{\leq g(N')}$ into $B_g(N')$. The checking operation then is very fast: when a node $N$ is generated, test membership of $s(N)$ in $B_g$ for all $g \leq g(N)$. Each such test takes time linear in the size of the state.

5 The Compositional Approach

As hinted, our approach is compositional, constructing the dominance relation over the state space $\Theta_{\Pi}$ as the composition of simulation relations over orthogonal projections thereof. Stating this in a generic manner (and simplifying to the atomic case of two orthogonal projections), we have an LTS $\Theta^2$ which equals the synchronized product $\Theta^1 \otimes \Theta^2$ of two smaller LTSs. We obtain a simulation for $\Theta^2$ from simulations for $\Theta^1$ and $\Theta^2$.

Definition 3 (Relation Composition) Let $\Theta^1 = (S^1, L, T^1, s_0^1, S_1^1)$ and $\Theta^2 = (S^2, L, T^2, s_0^2, S_2^2)$ be LTSs sharing the same labels. For binary relations $\preceq_1 \subseteq S_1 \times S_1$ and $\preceq_2 \subseteq S_2 \times S_2$, the composition of $\preceq_1$ and $\preceq_2$, denoted $\preceq_1 \otimes \preceq_2$, is the binary relation on $S_1 \times S_2$ where $(s_1, s_2) \in (\preceq_1 \otimes \preceq_2)(t_1, t_2)$ iff $s_1 \preceq_1 t_1$ and $s_2 \preceq_2 t_2$.

Proposition 1 Let $\Theta_{\Pi}^2 = \Theta^1 \otimes \Theta^2$, and let $\preceq_1$ and $\preceq_2$ be goal-respecting simulations for $\Theta^1$ and $\Theta^2$ respectively. Then $\preceq_1 \otimes \preceq_2$ is a goal-respecting simulation for $\Theta_{\Pi}^2$.

The proof is direct by definition, and is almost identical to that of a similar result concerning bisimulation, stated by Helmert et al. [2014].

Our basic idea can now be described as follows. Say we have a planning task $\Pi = (V, A, I, G)$ with state space $\Theta_{\Pi}$, a partition $V_1, \ldots, V_k$ of the task’s variables, and a goal-respecting bisimulation abstraction $\alpha_i$ of each $\tau(\Theta_{\Pi}^V_i)$ where $\tau$ is an exact label reduction. This is precisely the input we will get from merge-and-shrink abstraction. We will henceforth refer to this input as our initial abstractions. Say we construct a goal-respecting simulation $\preceq_i$ for each abstract state space $\Theta_{\Pi}^V_i$. Because bisimulation is a special case of simulation, $\preceq_i$ is a goal-respecting simulation for $\tau(\Theta_{\Pi}^V_i)$. Applying Definition 3 and Proposition 1 iteratively, $\otimes_i \preceq_i$ is a goal-respecting simulation for $\otimes_i \tau(\Theta_{\Pi}^V_i)$. Because $\tau$ is exact, $\otimes_i \tau(\Theta_{\Pi}^V_i) = \otimes_i \Theta_{\Pi}^V_i$, and by the reconstruction property $\otimes_i \Theta_{\Pi}^V_i = \tau(\Theta_{\Pi})$. Because the label reduction is cost-preserving, $\otimes_i \preceq_i$ is a cost-simulation for $\Theta_{\Pi}$, and hence a dominance relation as desired.

One can use this result and method as-is, obtaining a new dominance pruning method as a corollary of suitably assembling existing results and methods. However, empirically, this method’s ability to find interesting dominance relations is quite limited. We now extend the simulation concept to overcome that problem.

6 Label-Dominance Simulation

Sievers et al. [2014] introduce label “subsumption”, where $l'$ subsumes $l$ if it labels all transitions labeled by $l$. To intertwine dominance between labels with dominance between states, we extend that concept as follows:
**Definition 4 (Label Dominance)** Let Θ be an LTS with states S, let $s \preceq S \times S$ be any binary relation on $S$, and let $l, l'$ be labels. We say that $l'$ **dominates** $l$ in $Θ$ given $s \preceq s'$ if $c(l') \leq c(l)$, and for every transition $s \xrightarrow{l} s'$ there exists a transition $s \xrightarrow{l'} t$ s.t. $s' \preceq t'$.

The relation $\preceq$ here is arbitrary, but will be a simulation in practice. Hence, intuitively, a label dominates another one if it “applies to the same states and always leads to an at least as good state”. To give a simple example, consider the LTS corresponding to a single vehicle’s position, and say we have a 0-cost “beam” action which takes us from any position to the vehicle’s goal. Provided that every position is, per $\preceq$, simulated by the goal position, “beam” dominates all other labels.

In IPC benchmarks, typically Definition 4 is important not for regular actions, but for NOOPs. For illustration, say we have a truck variable $v_T$, two locations $A$ and $B$, and a package variable $v_P$ whose goal is to be at $B$. Our variable partition is the trivial one, $V_1 = \{v_T\}$ and $V_2 = \{v_P\}$. Bisimulation using exact label reduction will return LTSs as shown in Figure 1. The “load” and “unload” actions get reduced in a way allowing to synchronize with the correct truck position; the distinction between the truck “drive” actions is irrelevant so these are reduced to the same label. Clearly, no label dominates any other in either of these two LTSs. However, consider $Θ^1$ and $Θ^2$ with NOOPs added. The new label $noop$ dominates the load/unload actions in $Θ^1$, and dominates the drive actions in $Θ^2$, provided $\preceq_1$ and $\preceq_2$ are reflexive as will be the case in practice.

![Figure 1: Label-reduced bisimulations](image)

**Definition 5 (Label-Dominance Simulation)** Let $T = \{Θ^1, \ldots, Θ^k\}$ be a set of LTSs sharing the same labels. Denote the states of $Θ^i$ by $S_i$. A set $R = \{\preceq_1, \ldots, \preceq_k\}$ of binary relations $\preceq_i \subseteq S_i \times S_i$ is a label-dominance simulation for $T$ if, whenever $s \preceq_i t$, $s, t \in S_i^G$, 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 $R$ the coarsest label-dominance simulation if, for every label-dominance simulation $R'$ = $\{\preceq'_1, \ldots, \preceq'_k\}$ for $T$, we have $\preceq'_i \subseteq \preceq_i$ for all $i$.

A unique coarsest label-dominance simulation always exists, and can be computed in time polynomial in the size of $T$. We will prove this in the next section as a corollary of specifying our algorithm for doing this computation.

In the example, $T \preceq_2 B$ holds because $s \xrightarrow{noop} s'$ in $Θ^2$ is $T \xrightarrow{noop} B$, and the simulating $t \xrightarrow{\text{tmp}} t'$ in $Θ^2$ is $B \xrightarrow{\text{tmp}} B$, which works because $c(\text{noop}) = 0 \leq 1 = c(\text{IB})$, $B \preceq_2 B$, and $noop$ dominates IB in $Θ^1$. In the same fashion, provided that $A \preceq_2 B$, the transition $T \xrightarrow{\text{IB}} A$ is simulated by $B \xrightarrow{\text{IB}} B$. Note that neither of these two inferences could be made with the standard concept of simulation (even after exact label reduction), because that concept insists on using the same labels, not dominating ones.

We now prove soundness of label-dominance simulation, i.e., that label-dominance simulations $R$ yield cost-abstractions provided by merge-and-shrink abstraction, i.e., reducing the cost by $\sum c(l)$ of $l$.

**Lemma 1** Let $T = \{Θ^1, \ldots, Θ^k\}$ be a set of LTSs sharing the same labels, and let $R = \{\preceq_1, \ldots, \preceq_k\}$ be a label-dominance simulation for $T$. Then $\{\preceq_1 \otimes \preceq_2, \ldots, \preceq_k\}$ is a label-dominance simulation for $\{Θ^1 \otimes Θ^2, \ldots, Θ^k\}$.

**Proof Sketch**: The claim regarding $\preceq_3, \ldots, \preceq_k$ is simple. For $\preceq_1 \otimes \preceq_2$, consider states $(s_1, s_2) \preceq_1 (s'_1, s'_2)$ and a transition $(s_1, s_2) \xrightarrow{l} (s_1', s_2')$ as follows: 1. As $s_1 \preceq_1 t_1$, obtain a transition $s_1 \xrightarrow{\text{tmp}} s_1'$ dominating $s_1' \xrightarrow{l} s_1'$ in $Θ^1$. 2. As $l$ dominates $l'$ in $Θ^2$, obtain a transition $s_2 \xrightarrow{\text{tmp}} s_2'$ dominating $s_2' \xrightarrow{l'} s_2'$ in $Θ^2$. 3. As $s_2 \preceq_2 t_2$, obtain a transition $s_2 \xrightarrow{\text{tmp}} s_2'$ dominating $s_2' \xrightarrow{l'} s_2'$ in $Θ^2$. 4. As $l'$ dominates $l'$ in $Θ^1$, obtain a transition $t_1 \xrightarrow{\text{tmp}} t_1'$ dominating $t_1 \xrightarrow{\text{tmp}} t_1'$ in $Θ^1$.

**Theorem 1** Let $T = \{Θ^1, \ldots, Θ^k\}$ be a set of LTSs sharing the same labels, and let $R = \{\preceq_1, \ldots, \preceq_k\}$ be a label-dominance simulation for $T$. Then $\otimes_i \preceq_i$ is a cost-simulation for $\otimes_i Θ^i$.

**Proof**: Applying Lemma 1, we get that $\otimes_i \preceq_i$ is a label-dominance simulation for $\otimes_i Θ^i$. Now, for such a singleton set of LTSs, the requirements on the transition $t \xrightarrow{l'} t'$ replacing $s \xrightarrow{l} s'$ are that $c(l') \leq c(l)$, and $s' \preceq_i t'$. Hence label-dominance simulation simplifies to cost-simulation, and the claim follows.

To clarify the overall process, assume now again the initial abstractions provided by merge-and-shrink abstraction, i.e.,
a partition \(V_1, \ldots, V_k\) of the variables, and a goal-respecting bisimulation abstraction \(\alpha_i\), with abstract state space \(\Theta^{\alpha_i}\), of each \(\tau(\Theta^{\alpha_i}_i)\) where \(\tau\) is an exact label reduction. We compute the coarsest label-dominance simulation \(R = \{\leq_1, \ldots, \leq_k\}\) for \(T := \{\Theta^{\alpha_{i_{\text{noop}}}}_1, \ldots, \Theta^{\alpha_{i_{\text{noop}}}}_k\}\). As adding NOOPs does not affect bisimulation and is interchangeable with the synchronized product, with Theorem 1 we have that \(\otimes_i \leq_i\) is a cost-simulation for \(\Theta^{\alpha_i}_i\), and hence a cost-simulation for \(\Theta_{\text{II}}\) as desired.

7 Computing \(R\)

We now show how to operationalize Definition 5: Given \(T = \{\Theta^1, \ldots, \Theta^k\}\), how to compute the coarsest label-dominance simulation \(R\) for \(T\)?

It is well known that the coarsest simulation can be computed in time polynomial in the size of the input LTS [Henzinger et al., 1995]. The algorithm starts with the most generous relation \(\leq\) possible, then iteratively removes pairs \(s \leq t\) that do not satisfy the simulation condition. When no more changes occur, the unique coarsest simulation has been found. This method extends straightforwardly to label-dominance simulation.

Proposition 2 Let \(T = \{\Theta^1, \ldots, \Theta^k\}\) be a set of LTSs sharing the same labels. Then a unique coarsest label-dominance simulation for \(T\) exists.

Proof: The identity relation is a label-dominance simulation. If \(R = \{\leq_1, \ldots, \leq_k\}\) and \(R' = \{\leq'_1, \ldots, \leq'_k\}\) are label-dominance simulations, then \(\leq_1 \cup \leq'_1, \ldots, \leq_k \cup \leq'_k\) also is a label-dominance simulation.

Denote the states of \(\Theta^i\) by \(S_i\). Define the Boolean function \(O_k(i, s, t)\), where \(s \leq_i t\), to return \(\text{true}\) iff the condition for label-dominance simulation holds, i.e., iff \(s \in S_i^G\) implies that \(t \in S_i^G\), and for every transition \(s \xrightarrow{t} s'\) in \(\Theta^i\) there exists a transition \(t' \xrightarrow{l'} s'\) in \(\Theta^i\) such that \(c(l') \leq c(l)\), \(s' \leq_i t'\), and, for all \(j \neq i\), \(l' \) dominates \(l\) in \(\Theta^j\) given \(\leq_j\). Our algorithm proceeds as follows:

For all \(i\), set \(\leq_i := \{(s, t) \mid s \in S_i, s \notin S_G^c, t \notin S_G^c\}\)
while \(\text{ex. } (i, s, t)\) s.t. not \(O_k(i, s, t)\) do
Select one such triple \((i, s, t)\)
Set \(\leq_i := \leq_i \setminus \{(s, t)\}\)
endwhile
return \(R := \{\leq_1, \ldots, \leq_k\}\)

Proposition 3 Let \(T = \{\Theta^1, \ldots, \Theta^k\}\) be a set of LTSs sharing the same labels. Our algorithm terminates in time polynomial in the size of \(T\), and returns the coarsest label-dominance simulation for \(T\).

Proof: Each iteration reduces one \(\leq_i\) by one element. This gives a polynomial bound on the number of iterations, and every iteration takes polynomial time.

The returned \(R\) is a label-dominance simulation as that is the termination condition. \(R\) is coarsest as every label-dominance simulation must refine the initial relations \(\{\{(s, t) \mid s, t \in S_i, s \notin S_G^c, t \in S_G^c\}\}\), and every time we remove a pair \((s, t)\) we know that \(s \nleq_i t\) in any label-dominance simulation.

8 Experiments

Our techniques are implemented in Fast Downward (FD) [Helmer, 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 Intel E5-2660 machines running at 2.20 GHz, with time (memory) cut-offs of 30 minutes (4 GB). We run A* with FD’s blind heuristic, and with LM-cut [Helmer and Domshlak, 2009]. We perform an ablation study of label-dominance simulation, vs. standard simulation (neither NOOPs nor label-dominance), vs. bisimulation as computed by merge-and-shrink (not doing any work on top of merge-and-shrink, just using its output for the pruning). To represent the state of the art in alternative pruning methods, we include the best-performing partial-order reduction based on strong stubborn sets, which dominates other partial-order pruning approaches such as expansion-core [Wrehle et al., 2013].

Our initial abstractions are obtained using merge-and-shrink with exact label reduction, bisimulation shrinking, and the non-linear merge DFP strategy [Dräger et al., 2006; 2009; Sievers et al., 2014]. We impose two bounds on this process, namely a time limit of 300 seconds, as well as a limit \(M\) on the number of abstract transitions. When either of these limits is reached, the last completed abstractions form the starting point for our simulation process, i.e., are taken to be the initial abstractions. With this arrangement of parameters, the trade-off between merge-and-shrink overhead incurred vs. benefits gained is relatively easy to control. The bound on transitions works better than the more usual bound on abstract states, because the same number of abstract states may lead to widely differing numbers of transitions and thus actual effort. A reasonably good “magic” setting for \(M\), in our current context, is 100k. For \(M = 0\), i.e. computing the component simulations on individual state variables only, performance is substantially worse. For \(M = 200k\), the overhead becomes prohibitive. In between, overall coverage undergoes relatively small changes only (in the order of 5 instances).

Consider Table 1. With our pruning method, nodes are first generated and then checked for pruning, so the evaluated states are exactly the non-pruned generated ones. Hence the number of evaluated states assesses our pruning power, and the ratio between generated nodes and search time assesses the average time-per-node. The “safety belt” disables pruning if, after 1000 expansions, no node has been pruned. This is a simple yet effective method to avoid runtime overhead in cases where no or not much pruning will be obtained.

Compared to partial-order reduction, simulation-based pruning tends to be “stronger on its own, but less complementary to LM-cut”. Consider first the blind heuristic, which assesses the pruning power of each technique “on its own”. Simulation-based pruning typically yields much stronger evaluation reductions, the only clear exception being ParcPrinter where partial-order reduction excels. This results in much better coverage in many domains and overall. With LM-cut, on the other hand, while simulation-based pruning still applies more broadly – there are 14 test suites where it reduces evaluations but partial-order reduction does not – the extent of the reduction is dramatically diminished.
Table 1: Experiments. "A": A current implementation of that process is largely naïve. We overhead stems from computing the simulation relations. Our size reduction (compare the respective "Gen/sec." vs. "Evaluations") is the factor by which the per-domain summed-up number of evaluated coverage occur anywhere are omitted. "L": label-dominance simulation; "S": simulation; "B": bisimulation; "P": partial-order reduction. Domains where no changes in coverage occur anywhere are omitted. "Coverage" is the factor by which the per-domain summed-up number of evaluated states, relative to "A", decreases. "Gen/sec." is the factor by which the per-node runtime (summed-up number of generated nodes divided by summed-up search time), relative to "A", increases.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Airport</td>
<td>50</td>
<td>0.2</td>
<td>-7.4</td>
<td>-7.0</td>
<td>1.2</td>
<td>1.2</td>
<td>4.4</td>
<td>341.11</td>
<td>1.2</td>
</tr>
<tr>
<td>Driverlog</td>
<td>30</td>
<td>7</td>
<td>+2</td>
<td>+2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4.8</td>
<td>2.1</td>
</tr>
<tr>
<td>Elevators08</td>
<td>30</td>
<td>14</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>0.9</td>
<td>4.1</td>
</tr>
<tr>
<td>Elevators11</td>
<td>20</td>
<td>15</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>1.8</td>
<td>0.8</td>
</tr>
<tr>
<td>Floortile11</td>
<td>20</td>
<td>2</td>
<td>+4</td>
<td>+4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5.7</td>
<td>3.7</td>
</tr>
<tr>
<td>Floortile14</td>
<td>20</td>
<td>0</td>
<td>+5</td>
<td>+5</td>
<td>+5</td>
<td>0</td>
<td>0</td>
<td>6.3</td>
<td>1.1</td>
</tr>
<tr>
<td>FreeCell</td>
<td>20</td>
<td>7</td>
<td>-7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.3</td>
<td>1.6</td>
</tr>
<tr>
<td>Gripper</td>
<td>20</td>
<td>8</td>
<td>+6</td>
<td>+6</td>
<td>+6</td>
<td>0</td>
<td>0</td>
<td>10.3</td>
<td>3.1</td>
</tr>
<tr>
<td>Hiking14</td>
<td>20</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-3</td>
<td>1.8</td>
<td>3.1</td>
<td>30.5</td>
</tr>
<tr>
<td>Logistics08</td>
<td>20</td>
<td>10</td>
<td>+7</td>
<td>+6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9.3</td>
<td>1.1</td>
</tr>
<tr>
<td>Logistics98</td>
<td>20</td>
<td>24</td>
<td>+1</td>
<td>+1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4.4</td>
<td>0.3</td>
</tr>
<tr>
<td>Micomic</td>
<td>150</td>
<td>55</td>
<td>+6</td>
<td>+6</td>
<td>+1</td>
<td>-1</td>
<td>-1</td>
<td>15.6</td>
<td>5.5</td>
</tr>
<tr>
<td>Mprime</td>
<td>20</td>
<td>20</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>18.8</td>
<td>1.5</td>
</tr>
<tr>
<td>Mystery</td>
<td>30</td>
<td>15</td>
<td>-3</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29.2</td>
<td>14.2</td>
</tr>
<tr>
<td>NoMyystery</td>
<td>20</td>
<td>5</td>
<td>+10</td>
<td>+10</td>
<td>+1</td>
<td>0</td>
<td>0</td>
<td>46.4</td>
<td>23</td>
</tr>
<tr>
<td>OpenStack08</td>
<td>20</td>
<td>22</td>
<td>+2</td>
<td>+2</td>
<td>+2</td>
<td>0</td>
<td>0</td>
<td>21.0</td>
<td>9.3</td>
</tr>
<tr>
<td>OpenStack10</td>
<td>20</td>
<td>17</td>
<td>+2</td>
<td>+2</td>
<td>+2</td>
<td>0</td>
<td>0</td>
<td>7.8</td>
<td>9.3</td>
</tr>
<tr>
<td>PegNoTank</td>
<td>30</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>1</td>
<td>18.0</td>
<td>1.6</td>
</tr>
<tr>
<td>PipesNoTank</td>
<td>10</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.3</td>
<td>1.4</td>
</tr>
<tr>
<td>PipesTank</td>
<td>50</td>
<td>12</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16.8</td>
<td>25.2</td>
</tr>
<tr>
<td>Rovers</td>
<td>40</td>
<td>6</td>
<td>+2</td>
<td>+2</td>
<td>+1</td>
<td>0</td>
<td>0</td>
<td>20.6</td>
<td>3.7</td>
</tr>
<tr>
<td>Satellite</td>
<td>30</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8.4</td>
<td>3.8</td>
</tr>
<tr>
<td>ScalyzYzer08</td>
<td>20</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>11.0</td>
<td>10.3</td>
</tr>
<tr>
<td>ScalyzYzer10</td>
<td>20</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.1</td>
<td>1.6</td>
</tr>
<tr>
<td>Sokoban08</td>
<td>30</td>
<td>22</td>
<td>-9</td>
<td>-9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.7</td>
<td>8.2</td>
</tr>
<tr>
<td>Sokoban10</td>
<td>30</td>
<td>19</td>
<td>-9</td>
<td>-9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.6</td>
<td>8.1</td>
</tr>
<tr>
<td>Tetris</td>
<td>10</td>
<td>14</td>
<td>-6</td>
<td>-6</td>
<td>+1</td>
<td>0</td>
<td>0</td>
<td>5.2</td>
<td>5.2</td>
</tr>
<tr>
<td>Tidybot10</td>
<td>50</td>
<td>9</td>
<td>-8</td>
<td>-7</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>59.4</td>
<td>6.5</td>
</tr>
<tr>
<td>Tidybot14</td>
<td>20</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>9.7</td>
<td>3.9</td>
</tr>
<tr>
<td>TTP</td>
<td>30</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>22.7</td>
<td>3.3</td>
</tr>
<tr>
<td>Transport14</td>
<td>20</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1.9</td>
<td>1.4</td>
</tr>
<tr>
<td>Trucks</td>
<td>30</td>
<td>6</td>
<td>+2</td>
<td>+2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13.8</td>
<td>6</td>
</tr>
<tr>
<td>VisitAll11</td>
<td>20</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10.4</td>
<td>3.5</td>
</tr>
<tr>
<td>VisitAll14</td>
<td>20</td>
<td>3</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>0</td>
<td>0</td>
<td>9.2</td>
<td>3.5</td>
</tr>
<tr>
<td>Woodwork08</td>
<td>30</td>
<td>8</td>
<td>+10</td>
<td>+10</td>
<td>+5</td>
<td>+5</td>
<td>+5</td>
<td>91.4</td>
<td>23.7</td>
</tr>
<tr>
<td>Woodwork11</td>
<td>20</td>
<td>3</td>
<td>+9</td>
<td>+9</td>
<td>+5</td>
<td>+6</td>
<td>+6</td>
<td>91.6</td>
<td>23.8</td>
</tr>
<tr>
<td>Zenotravel</td>
<td>20</td>
<td>8</td>
<td>+1</td>
<td>+1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4.3</td>
<td>6.2</td>
</tr>
</tbody>
</table>

\[ \sum \text{gen/sec.} = 1271 \]

\[ \sum \text{gen/sec. (A)} = 605 +19 +57 +16 +16 +8 = 833 +3 +20 +14 +17 +38 \]

Partial-order reduction suffers from this as well, but retains much of its power in Parsec and Woodworking, and consistently causes very little runtime overhead relative to this slow heuristic function. Thus partial-order reduction has better overall coverage. It does not dominate simulation-based pruning though, which yields better coverage in Floortile, Gripper, NoMyystery, TTP, and VisitAll.

Label-dominance simulation clearly pays off against standard simulation as well as bisimulation. The latter already is very helpful in some domains, like Gripper and Woodworking. Simulation does add over this, but suffers in some domains, like Tidybot, from the runtime overhead. Label-dominance simulation has such issues as well, but makes up for them by more pronounced gains on other domains.

The per-node runtime overhead in simulation-based pruning is almost consistently outweighed by the search space size reduction (compare the respective "Gen/sec." vs. "Evaluations" columns in Table 1). The most substantial runtime overhead stems from computing the simulation relations. Our current implementation of that process is largely naïve. We experimented with ideas from model checking for doing this more effectively, but with limited success due to the different context (especially, label-dominance). It remains an important open topic to improve this part of our machinery.

9 Conclusion

The idea of pruning states based on some form of “dominance” is old, but has previously been incarnated in planning with simple special cases (“more facts true”, “more resources available”) only. Simulation relations are the natural framework to move beyond this. Our work constitutes a first step towards leveraging the power of simulation relations in, as well as extending them for, admissible pruning in planning. The method is orthogonal to existing pruning methods, and empirically exhibits complementary strengths relative to partial-order reduction, so there is potential for synergy. A major challenge in our view is how to intelligently control initial-abstraction size, investing a lot of overhead where simulation pruning is promising and, ideally, avoiding any overhead altogether where it is not.
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


