HAL
open science

How to Relax a Bisimulation?

Michael Katz, Joerg Hoffmann, Malte Helmert

To cite this version:

Michael Katz, Joerg Hoffmann, Malte Helmert. How to Relax a Bisimulation?. 22nd International Conference on Automated Planning and Scheduling (ICAPS), Jun 2012, Atibaia, Brazil. hal-00765027

HAL Id: hal-00765027

https://inria.hal.science/hal-00765027

Submitted on 14 Dec 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

How to Relax a Bisimulation?

Michael Katz and **Jörg Hoffmann**
Saarland University
Saarbrücken, Germany
{katz, hoffmann}@cs.uni-saarland.de

Malte Helmert
University of Basel
Basel, Switzerland
malte.helmert@unibas.ch

Abstract

Merge-and-shrink abstraction (M&S) is an approach for constructing admissible heuristic functions for cost-optimal planning. It enables the targeted design of abstractions, by allowing to choose individual pairs of (abstract) states to aggregate into one. A key question is how to actually make these choices, so as to obtain an informed heuristic at reasonable computational cost. Recent work has addressed this via the well-known notion of *bisimulation*. When aggregating only bisimilar states – essentially, states whose behavior is identical under every planning operator – M&S yields a perfect heuristic. However, bisimulations are typically exponentially large. Thus we must *relax* the bisimulation criterion, so that it applies to more state pairs, and yields smaller abstractions. We herein devise a fine-grained method for doing so. We restrict the bisimulation criterion to consider only a subset K of the planning operators. We show that, if K is chosen appropriately, then M&S still yields a perfect heuristic, while abstraction size may decrease exponentially. Designing practical approximations for K, we obtain M&S heuristics that are competitive with the state of the art.

Introduction

Heuristic forward state-space search with A* and admissible heuristics is a state of the art approach to cost-optimal domain-independent planning. The main research question in this area is how to derive the heuristic automatically. That is what we contribute to herein. We design new variants of the merge-and-shrink heuristic, short M&S, whose previous variant (Nissim, Hoffmann, and Helmert 2011b) won a 2nd price in the optimal planning track of the 2011 International Planning Competition (IPC), and was part of the 1st-prize winning portfolio (Helmert et al. 2011).

M&S uses solution cost in a smaller, *abstract* state space to yield an admissible heuristic. The abstract state space is built incrementally, starting with a set of atomic abstractions corresponding to individual variables, then iteratively *merging* two abstractions (replacing them with their synchronized product) and *shrinking* them (aggregating pairs of states into one). In this way, M&S allows to select individual pairs of (abstract) states to aggregate. A key question, that governs both the computational effort taken and the quality of the resulting heuristic, is *how to actually select these state pairs*.

Copyright © 2012, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

M&S was first introduced for planning by Helmert et al. (2007), with only a rather naïve method for selecting the state pairs to aggregate. Nissim et al. (2011a) more recently addressed this via the notion of *bisimulation*, adopted from the verification literature (e. g., (Milner 1990)). Two states s, t are bisimilar, roughly speaking, if every transition label (every planning operator) leads into equivalent abstract states from s and t. If one aggregates only bisimilar states, then the behavior of the transition system (the possible paths) remains unchanged. This property is invariant over both the merging and shrinking steps in M&S, and thus the resulting heuristic is guaranteed to be perfect. Unfortunately, bisimulations are exponentially big even in trivial examples, including benchmarks like, for example, Gripper.

A key observation made by Nissim et al. is that bisimulation is unnecessarily strict for our purposes. In verification, paths must be preserved because the to-be-verified property shall be checked within the abstracted system. However, here we only want to compute solution costs. Thus it suffices to preserve not the actual paths, *but only their cost*. Nissim et al. design a label reduction technique, that changes the path inscriptions (the associated planning operators) but not their costs. This leads to polynomial behavior in Gripper and some other cases, but the resulting abstractions are still much too large in most planning benchmarks.

Nissim et al. address this by (informally) introducing what they call *greedy bisimulation*, which "catches" only a subset of the transitions: s, t are considered bisimilar already if every transition *decreasing remaining cost* leads into equivalent abstract states from s and t. That is, "bad transitions" – those increasing remaining cost – are ignored. This is a *lossy* relaxation to bisimulation, i. e., a simplification that results in smaller abstractions but may (and usually does) yield an imperfect heuristic in M&S: "bad transition" is defined locally, relative to the current abstraction, which does not imply that the transition is globally bad. For example, driving a truck away from its own goal may be beneficial for transporting a package. Under such (very common) behavior, greedy bisimulation is not invariant across the M&S merging step, because the relevant transitions are not caught.

We herein adopt the same approach for relaxing bisimulation – we catch a subset of the transitions – but we take a different stance for determining that subset. We first select a subset of labels (operators). Then, throughout M&S,

we catch the transitions bearing these labels. This simple technique warrants that the thus-relaxed bisimulation is invariant across M&S. Thanks to this, to guarantee a quality property ϕ of the final M&S heuristic, it suffices to select a label subset guaranteeing ϕ when catching these labels in a bisimulation of the (global) state space.

We consider two properties ϕ: (A) obtaining a perfect heuristic; (B) guaranteeing that A* will not have to search. (A) is warranted by selecting all remaining-cost decreasing operators in the (global) state space. (B) is a generalization that only requires to catch a subset of these operators – those within a certain radius around the goal.

In practice, it is not feasible to compute the label sets just described. To evaluate their potential in principle, we prove that they may decrease abstraction size exponentially, and we run experiments on IPC benchmarks small enough to determine these labels. To evaluate the potential in practice, we design approximation methods. Running these on the full IPC benchmarks, we establish that the resulting M&S heuristics are competitive with the state of the art, and can improve coverage in some domains.

For space reasons, we omit many details. Full details are available in a TR (Katz, Hoffmann, and Helmert 2012).

Background

A **planning task** is a 4-tuple $\Pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$. $\mathcal{V}$ is a finite set of **variables** v, each $v \in \mathcal{V}$ associated with a finite domain $\mathcal{D}_v$. A **partial state** over $\mathcal{V}$ is a function s on a subset $\mathcal{V}_s$ of $\mathcal{V}$, so that $s(v) \in \mathcal{D}_v$ for all $v \in \mathcal{V}_s$; s is a **state** if $\mathcal{V}_s = \mathcal{V}$. The **initial state** s_0 is a state. The **goal** $s_\star$ is a partial state. $\mathcal{O}$ is a finite set of **operators**, each being a pair $(\mathit{pre}, \mathit{eff})$ of partial states, called its **precondition** and **effect**. Each $o \in \mathcal{O}$ is also associated with its **cost** $c(o) \in \mathbb{R}_0^+$ (note that 0-cost operators are allowed). A special case we will mention are **uniform** costs, where $c(o) = 1$ for all o.

The semantics of planning tasks are defined via their **state spaces**, which are (labeled) **transition systems**. Such a system is a 5-tuple $\Theta = (S, L, T, s_0, S_\star)$ where S is a finite set of **states**, L is a finite set of **transition 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 $S_\star \subseteq S$ is the set of **solution states**. We define the **remaining cost** $h^* : S \to \mathbb{R}_0^+$ as the minimal cost of any path (the sum of costs of the labels on the path), in Θ, from a given state s to any $s_\star \in S_\star$, or $h^*(s) = \infty$ if there is no such path.

In the state space of a planning task, S is the set of all states. The start state s_0 is the initial state of the task, and $s \in S_\star$ if $s_\star \subseteq s$. The transition labels L are the operators $\mathcal{O}$, and $(s, (\mathit{pre}, \mathit{eff}), s') \in T$ if s complies with *pre*, and $s'(v) = \mathit{eff}(v)$ for all $v \in \mathcal{V}_{\mathit{eff}}$ while $s'(v) = s(v)$ for all $v \in \mathcal{V} \setminus \mathcal{V}_{\mathit{eff}}$. A **plan** is a path from s_0 to any $s_\star \in S_\star$. The plan is **optimal** iff its summed-up cost is equal to $h^*(s_0)$.

A **heuristic** is a function $h : S \to \mathbb{R}_0^+ \cup \{\infty\}$. The heuristic is **admissible** iff, for every $s \in S$, $h(s) \leq h^*(s)$; it is **consistent** iff, for every $(s, l, s') \in T$, $h(s) \leq h(s') + c(l)$; it is **perfect** iff h coincides with h^*. The A* algorithm expands states by increasing value of $g(s) + h(s)$ where $g(s)$ is the accumulated cost on the path to s. If h is admissible, then A* returns an optimal solution. If h is consistent then A* does not need to re-open any nodes. If h is perfect then, as will be detailed later, A* "does not need to search"; we will also identify a more general criterion sufficient to achieve this last property.

How to automatically compute a heuristic, given a planning task as input? Our approach is based on designing an **abstraction**. This is a function α mapping S to a set of **abstract states** S^α. The **abstract state space** Θ^α is defined as $(S^\alpha, L, T^\alpha, s_0^\alpha, S_\star^\alpha)$, where $T^\alpha := \{(\alpha(s), l, \alpha(s')) \mid (s, l, s') \in T\}$, $s_0^\alpha := \alpha(s_0)$, and $S_\star^\alpha := \{\alpha(s_\star) \mid s_\star \in S_\star\}$. The **abstraction heuristic** h^α maps each $s \in S$ to the remaining cost of $\alpha(s)$ in Θ^α; h^α is admissible and consistent. We will sometimes consider the **induced equivalence relation** $\sim^\alpha$, defined by setting $s \sim^\alpha t$ iff $\alpha(s) = \alpha(t)$.

How to choose a good α in general? Inspired by work in the context of model checking automata networks (Dräger, Finkbeiner, and Podelski 2006), Helmert et al. (2007) propose M&S abstraction as a method allowing fine-grained abstraction design, selecting individual pairs of (abstract) states to aggregate. The approach builds the abstraction in an incremental fashion, iterating between *merging* and *shrinking* steps. In detail, an abstraction α is an **M&S abstraction over** $V \subseteq \mathcal{V}$ if it can be constructed using these rules:

(i) For $v \in \mathcal{V}$, $\pi_{\{v\}}$ is an M&S abstraction over $\{v\}$.

(ii) If β is an M&S abstraction over V and γ is a function on S^β, then $\gamma \circ \beta$ is an M&S abstraction over V.

(iii) If α_1 and α_2 are M&S abstractions over disjoint sets V_1 and V_2, then $\alpha_1 \otimes \alpha_2$ is an M&S abstraction over $V_1 \cup V_2$.

Rule (i) allows to start from **atomic projections**. These are simple abstractions $\pi_{\{v\}}$ (also written π_v) mapping each state $s \in S$ to the value of one selected variable v. **Rule (ii)**, the **shrinking step**, allows to iteratively aggregate an arbitrary number of state pairs, in abstraction β. Formally, this simply means to apply an additional abstraction γ to the image of β. In **rule (iii)**, the **merging step**, the merged abstraction $\alpha_1 \otimes \alpha_2$ is defined by $(\alpha_1 \otimes \alpha_2)(s) := (\alpha_1(s), \alpha_2(s))$.[1]

The above defines how to construct the abstraction α, but not how to actually compute the abstraction heuristic h^α. For that computation, the constraint $V_1 \cap V_2 = \emptyset$ in rule (iii) is important. While designing α, we maintain also the abstract state space Θ^α. This is trivial for rules (i) and (ii), but is a bit tricky for rule (iii). We need to compute the abstract state space $\Theta^{\alpha_1 \otimes \alpha_2}$ of $\alpha_1 \otimes \alpha_2$, based on the abstract state spaces Θ^{α_1} and Θ^{α_2} computed (inductively) for α_1 and α_2 beforehand. We do so by forming the **synchronized product** $\Theta^{\alpha_1} \otimes \Theta^{\alpha_2}$. This is a standard operation, its state space being $S^{\alpha_1} \times S^{\alpha_2}$, with a transition from (s_1, s_2) to (s_1', s_2') via label l iff $(s_1, l, s_1') \in T^{\alpha_1}$ and $(s_2, l, s_2') \in T^{\alpha_2}$. As Helmert et al. (2007) show, the constraint $V_1 \cap V_2 = \emptyset$ is sufficient (and, in general, necessary) to ensure that this is correct, i. e., that $\Theta^{\alpha_1} \otimes \Theta^{\alpha_2} = \Theta^{\alpha_1 \otimes \alpha_2}$.

[1] Note that M&S abstractions are constructed over subsets V of the variables $\mathcal{V}$. Indeed, in practice, there is no need to incorporate all variables. Like the previous work on M&S, we do not make use of this possibility: all M&S abstractions in our experiments are over the full set of variables $V = \mathcal{V}$.

To implement M&S in practice, we need a **merging strategy** deciding which abstractions to merge in (iii), and a **shrinking strategy** deciding which (and how many) states to aggregate in (ii). Throughout this paper, we use the same merging strategy as the most recent work on M&S (Nissim, Hoffmann, and Helmert 2011a). What we investigate is the shrinking strategy. Helmert et al. (2007) proposed a strategy that leaves remaining cost intact within the current abstraction. This is done simply by not aggregating states whose remaining cost differs. The issue with this is that it preserves h^* *locally* only. For example, in a transportation domain, if we consider only the position of a truck, then any states s, t equally distant from the truck's target position can be aggregated: locally, the difference is irrelevant. *Globally,* however, there are transportable objects to which the difference in truck positions does matter, and thus aggregating s and t results in information loss.

We need a shrinking strategy that takes into account the global effect of state aggregations. Nissim et al. (2011a) address this via the well-known notion of *bisimulation*, a criterion under which an abstraction preserves exactly the behavior (the transition paths) of the original system:

Definition 1 *Let $\Theta = (S, L, T, s_0, S_\star)$ be a transition system. An equivalence relation $\sim$ on S is a* ***bisimulation*** *for Θ if $s \sim t$ implies that: (1) either $s, t \in S_\star$ or $s, t \notin S_\star$; (2) for every transition label $l \in L$, $\{[s'] \mid (s, l, s') \in T\} = \{[t'] \mid (t, l, t') \in T\}$.*

As usual, $[s]$ for a state s denotes the equivalence class of s. Intuitively, $s \sim t$ only if (1) s and t agree on the status of the goal, and (2) whatever operator applies to s or t applies to both, and leads into equivalent states. An abstraction α is a bisimulation iff the induced equivalence relation $\sim^\alpha$ is.

Note that there are potentially many bisimulations. For example, the identity relation, where $[s] = \{s\}$, is one. A bisimulation $\sim'$ is **coarser than** another bisimulation $\sim$ if $\sim' \supseteq \sim$, i. e., if every pair of states equivalent under $\sim$ is also equivalent under $\sim'$. A unique coarsest bisimulation always exists, and can be computed efficiently based on an explicit representation of Θ (Milner 1990). Thus the proposed shrinking strategy is to reduce, in any application of rule (ii), Θ^β to a coarsest bisimulation of itself.

It is easy to see that the bisimulation property is invariant over merging and shrinking steps. We spell out the claim for merging steps since we will generalize this result later on:[2]

Lemma 1 *(Nissim, Hoffmann, and Helmert 2011a) Let Θ_1 and Θ_2 be transition systems, and let α_1 and α_2 be abstractions for Θ_1 and Θ_2 respectively. If α_1 is a bisimulation for Θ_1, and α_2 is a bisimulation for Θ_2, then $\alpha_1 \otimes \alpha_2$ is a bisimulation for $\Theta_1 \otimes \Theta_2$.*

Proof sketch: For all $(s_1, s_2) \sim^{\alpha_1 \otimes \alpha_2} (t_1, t_2)$ in $\Theta_1 \otimes \Theta_2$, and all labels l, we need $\{[(s'_1, s'_2)] \mid ((s_1, s_2), l, (s'_1, s'_2)) \in T\} = \{[(t'_1, t'_2)] \mid ((t_1, t_2), l, (t'_1, t'_2)) \in T\}$. This follows directly by definition of $\Theta_1 \otimes \Theta_2$ and the prerequisites $\{[s'_i] \mid (s_i, l, s'_i) \in T_i\} = \{[t'_i] \mid (t_i, l, t'_i) \in T_i\}$. ■

[2]Note the slight abuse of notation here: α_i is a function on Θ_i, not on $\Theta_1 \otimes \Theta_2$; the precise claim is that $\overline{\alpha_1} \otimes \overline{\alpha_2}$ is a bisimulation for $\Theta_1 \otimes \Theta_2$, where $\overline{\alpha_1}(s_1, s_2) := \alpha_1(s_1)$ and $\overline{\alpha_2}(s_1, s_2) := \alpha_2(s_2)$. We omit this distinction to avoid notational clutter.

In other words, if we combine bisimulations for two transition systems, then we obtain a bisimulation for the synchronization of these systems. Due to this invariance property, bisimulation gets preserved throughout M&S: if we build an M&S abstraction α over the entire variable set $\mathcal{V} = \{v_1, \dots, v_n\}$, and we always shrink by coarsest bisimulation, then the abstraction will be a bisimulation for $\Theta^{\pi_{v_1}} \otimes \cdots \otimes \Theta^{\pi_{v_n}}$. The latter is isomorphic to the global state space (Helmert, Haslum, and Hoffmann 2007), thus α is a bisimulation for the global state space. Since bisimulation preserves transition paths exactly, this implies that h^α is a perfect heuristic (see the proof of Lemma 2 below).

As previously discussed, bisimulations are exponentially big even in trivial examples. A key point for improving on this is that, in contrast to verification where bisimulation is traditionally being used, we need to preserve not the solutions but only their cost. Nissim et al. (2011a) define a label reduction technique to this end, the details of which are not important here (we do use it in our implementation). What is important is that, even with the label reduction, in most benchmarks the resulting abstractions are still huge. The way out we employ here is to relax Definition 1 by applying constraint (2) to only a subset of the transitions in T – by **catching** this transition subset, as we will say from now on. We will show that this can be done while still computing a perfect heuristic, provided we catch the right transitions.

Nissim et al. already mentioned an approach – "greedy bisimulation"– catching a transition subset. The approach preserves h^* locally (in the current abstraction), but not globally. We next revisit it, then we introduce new techniques that preserve h^* globally.

Greedy Bisimulation

Nissim et al. propose greedy bisimulation as a more approximate shrinking strategy in practice, accepting its lack of global foresight. They introduce the concept informally only; formally, it is defined as follows:

Definition 2 *(Nissim, Hoffmann, and Helmert 2011a) Let $\Theta = (S, L, T, s_0, S_\star)$ be a transition system. An equivalence relation $\sim$ on S is a* ***greedy bisimulation*** *for Θ if it is a bisimulation for the system $(S, L, T^G, s_0, S_\star)$ where $T^G = \{(s, l, s') \mid (s, l, s') \in T, h^*(s') \leq h^*(s)\}$.*

In other words, greedy bisimulation differs from bisimulation in that it catches only the transitions not increasing remaining cost. Since a greedy bisimulation is a bisimulation on a modified transition system, it is obvious that a unique coarsest greedy bisimulation still exists and can be computed efficiently. More interestingly, greedy bisimulation still preserves (local) remaining cost:

Lemma 2 *Let Θ be a transition system, and let α be a greedy bisimulation for Θ. Then h^α is perfect.*

Proof sketch: We first show that any full bisimulation yields perfect heuristics. Say that (A, l, A') starts a cheapest abstract solution for $[s] = A$. By definition of the abstract transition system, there exists a transition $(t, l, t') \in T$ where $[t] = A$ and $[t'] = A'$. By Definition 1 (2), we have a transition (s, l, s') in Θ so that $s' \in [t'] = A'$. Thus the abstract

plan step has a real correspondence in the state s at hand. Iterating the argument yields, with Definition 1 (1), a real solution path with the same cost.

Next we show that $h^* = h^G$, where h^G denotes remaining cost in $\Theta^G = (S, L, T^G, s_0, S_\star)$. Removing transitions can only increase the remaining cost, so $h^* \leq h^G$. On the other hand, any *optimal* solution path in T is a solution path in T^G, thus $h^* \geq h^G$ as desired.

Now, let h' be the heuristic function defined as optimal solution cost in the quotient system Θ^G/α. Since α is a bisimulation of Θ^G, with the above we have $h' = h^G$. It thus suffices to show that $h^\alpha = h'$. That is the case because h^α can be obtained by adding, to Θ^G/α, all abstract transitions corresponding to $T \setminus T^G$: by construction, each added transition leads towards an abstract state with strictly greater abstract cost, so these costs remain the same. ■

The bad news, as indicated, is that remaining costs are not preserved at the *global* level. Say our shrinking strategy is to reduce, in any application of rule (ii), Θ^β to a coarsest greedy bisimulation of itself. Then, in difference to full bisimulation as per Definition 1, the final abstraction is *not* guaranteed to be a greedy bisimulation for the global state space. That is because greedy bisimulation is not invariant over merging steps, i. e., there is no equivalent of Lemma 1: a greedy bisimulation for Θ_1 does not catch transitions t that increase (local) remaining cost in Θ_1, however such t may *decrease* (global) remaining cost in $\Theta_1 \otimes \Theta_2$. A simple example is that where Θ_1 is a truck, Θ_2 is a package, and t drives the truck away from its own goal – which globally is a good idea in order to transport the package.

Not being invariant across M&S does not, by itself, imply that greedy bisimulation cannot result in useful heuristic functions in practice. Still, its unpredictable global effect is undesirable. And anyhow, greedy bisimulation actually catches more transitions than needed to preserve *local* remaining cost. We now introduce techniques addressing both.

Catching Relevant Labels

Instead of catching individual transitions with a criterion local to the current abstraction, we now devise techniques that catch them based on a label subset that we fix, with a global criterion, at the very beginning. Throughout the M&S process, we catch a transition iff its label is inside this subset.[3]

We next show that such label-catching bisimulations are invariant in M&S. We then define a subset of labels catching which guarantees a perfect heuristic. Subsequently, we show how this label subset can be further diminished, while still guaranteeing that A* will terminate without any search.

Catching Label Subsets

Definition 3 *Let $\Theta = (S, L, T, s_0, S_\star)$ be a transition system, and let K be a set of labels. An equivalence relation $\sim$ on S is a **K-catching bisimulation** for Θ if it is a bisimulation for the system $(S, K, T^K, s_0, S_\star)$ where $T^K = \{(s, l, s') \mid (s, l, s') \in T, l \in K\}$.*

[3]There is an interaction between "catching" labels, and "reducing" them as proposed by Nissim et al. (2011a). We do not reduce l and l' to the same label if l is caught but l' is not.

As indicated, K-catching bisimulation is invariant over M&S rule (iii), i. e., we can generalize Lemma 1 as follows:

Lemma 3 *Let Θ_1 and Θ_2 be transition systems, let K be a set of labels, and let α_1 and α_2 be abstractions for Θ_1 and Θ_2 respectively. If α_1 is a K-catching bisimulation for Θ_1, and α_2 is a K-catching bisimulation for Θ_2, then $\alpha_1 \otimes \alpha_2$ is a K-catching bisimulation for $\Theta_1 \otimes \Theta_2$.*

Proof sketch: For all $(s_1, s_2) \sim^{\alpha_1 \otimes \alpha_2} (t_1, t_2)$, we need $\{[(s'_1, s'_2)] \mid ((s_1, s_2), l, (s'_1, s'_2)) \in T, l \in K\} = \{[(t'_1, t'_2)] \mid ((t_1, t_2), l, (t'_1, t'_2)) \in T, l \in K\}$. As in Lemma 1, this follows by definition and prerequisites. ■

Thus we get invariance over the entire M&S process:

Lemma 4 *Let Π be a planning task with variables $\mathcal{V}$ and state space Θ, and let K be a set of labels. Let α be an M&S abstraction over $\mathcal{V}$ where, in any application of rule (ii), γ is a K-catching bisimulation for Θ^β. Then α is a K-catching bisimulation for Θ.*

Proof: Follows from Lemma 3 and the simple observations that the atomic M&S abstractions as per rule (i) are K-catching bisimulations, and that K-catching bisimulation is invariant over nested applications of rule (ii). ■

Catching a smaller label set can only decrease abstraction size, and can only increase the error made by the heuristic:

Lemma 5 *Let Θ be a transition system, and let $K' \subseteq K$ be sets of labels. Then the coarsest K'-catching bisimulation is coarser than the coarsest K-catching bisimulation.*

Proof: Denoting the coarsest K'-catching (K-catching) bisimulation with $\sim^{K'}$ ($\sim^K$), we need that $s \sim^K t$ implies $s \sim^{K'} t$. This holds because $\{[s'_i] \mid (s_i, l, s'_i) \in T, l \in K\} = \{[t'_i] \mid (t_i, l, t'_i) \in T, l \in K\}$ implies $\{[s'_i] \mid (s_i, l, s'_i) \in T, l \in K'\} = \{[t'_i] \mid (t_i, l, t'_i) \in T, l \in K'\}$. ■

Globally Relevant Labels

We now employ an idea similar to that of Nissim et al.'s greedy bisimulation, except that we select the transitions based on a global view, and a little more carefully:[4]

Definition 4 *Let Π be a planning task with state space $\Theta = (S, L, T, s_0, S_\star)$. A label $l \in L$ is **globally relevant** if there exists $(s, l, s') \in T$ such that $h^*(s') + c(l) = h^*(s)$.*

The transitions caught by this definition differ from those of Definition 2 in that (A) we identify them via their labels, rather than individually; (B) they refer to the global state space, not to the local abstraction; and (C) we do not catch transitions whose own cost exceeds the reduction in h^*. (A) is important to obtain invariance in M&S, as just discussed. (B) is needed because the global state space is what we wish to approximate. (C) suffices to obtain a perfect heuristic:

[4]In this definition, S and T (as defined in the background) include states not reachable from s_0. This is because, during M&S, reachability is over-approximated. If we do not catch the respective labels, then the abstraction is done on a transition system larger than that based on which we collected the labels, which may result in an imperfect heuristic even on reachable states. Our TR contains an example illustrating this phenomenon.

Lemma 6 *Let Π be a planning task with state space Θ, let G be the globally relevant labels, and let $K \supseteq G$. Let α be a K-catching bisimulation for Θ. Then h^α is perfect.*

Proof sketch: Due to Lemma 5, it suffices to consider the case $K = G$. The proof of Lemma 2 remains valid except in two details. When proving that $h^* = h^G$, we now rely on $h^*(s') + c(l) > h^*(s)$ to show that transitions not in T^G do not take part in optimal solution paths. Similarly when proving that $h' = h^\alpha$. ∎

Combining Lemmas 4 and 6, we get the desired result:

Theorem 1 *Let Π be a planning task with variables $\mathcal{V}$ and state space Θ, let G be the globally relevant labels, and let $K \supseteq G$. Let α be an M&S abstraction over $\mathcal{V}$ where, in any application of rule (ii), γ is a K-catching bisimulation for Θ^β. Then h^α is perfect.*

If there are no 0-cost operators, then with perfect h^α A* does not need to search. Precisely, A* finds an optimal plan after expanding a number of nodes linear in the plan's length, provided we break ties in A* based on smaller h^α. That is, if $g(s)+h^\alpha(s) = g(s')+h^\alpha(s')$ and $h^\alpha(s) < h^\alpha(s')$, then we expand s prior to s'. Given this, we know that (I) any state s' not on an optimal plan has $g(s')+h^\alpha(s') > g(s_0)+h^\alpha(s_0)$; and (II) along the states on any optimal plan, h^α decreases strictly monotonically. Due to (I), we do not expand any sub-optimal states. Due to (II), within the set of optimal states (which may be large), the tie-breaking leads directly to the goal in depth-first manner.

In the presence of 0-cost operators, (II) is no longer true, and in general there is no way to guarantee avoiding search (e. g., if *all* costs are 0 and h^* is devoid of information).

Bounded-Radius Relevant Labels

To avoid search in A*, it is not necessary for the heuristic to be perfect everywhere. It suffices to guarantee the conditions (I) and (II) above. We show that, to accomplish this, we can consider a radius R around the goal:

Definition 5 *Let Π be a planning task with state space $\Theta = (S, L, T, s_0, S_\star)$, and let $R \in \mathbb{R}_0^+$. A label $l \in L$ is **R-relevant** if there exists $(s, l, s') \in T$ such that $h^*(s')+c(l) = h^*(s) \leq R$.*

This "radius" in terms of a label subset translates into a radius guaranteeing heuristic quality:

Lemma 7 *Let Π be a planning task with state space Θ, let $R \in \mathbb{R}_0^+$, let G be the R-relevant labels, and let $K \supseteq G$. Let α be a K-catching bisimulation for Θ. Then, for every $s \in S$ with $h^*(s) \leq R$, we have $h^\alpha(s) = h^*(s)$; and for $s \in S$ with $h^*(s) > R$, we have $h^\alpha(s) > R$.*

Proof sketch: By a minor extension of the proof to Lemma 6. For transitions (s, l, s') with $h^*(s) \leq R$ the claim holds exactly as in Lemma 6. For transitions (s, l, s') with $h^*(s) > R$, if $h'([s']) \leq R$, then $h'([s']) = h^*(s')$, giving us $h'([s']) + c(l) \geq h^*(s) > R$; if $h'([s']) > R$, then adding (s, l, s') could never decrease $h'([s])$ below R. ∎

Combining this with Lemma 4 we get that, if we fix a label subset K catching all R-relevant labels, and if we implement the shrinking step as coarsest K-catching bisimulation, then the resulting heuristic h^α will have the claimed quality *on the global state space*. Thus, in the absence of 0-cost operators and when setting R to optimal plan cost, conditions (I) and (II) still hold, and A* is efficient:

Theorem 2 *Let Π be a planning task all of whose operators have non-0 cost. Let $\mathcal{V}$ be the variables of Π, let Θ be the state space of Π, let G be the $h^*(s_0)$-relevant labels, and let $K \supseteq G$. Let α be an M&S abstraction over $\mathcal{V}$ where, in any application of rule (ii), γ is a K-catching bisimulation for Θ^β. Then A* with h^α, breaking ties in favor of smaller heuristic values, expands a number of states linear in the length of the plan returned.*

Results Using Exact Label Sets

The label subsets introduced in the previous section cannot be computed efficiently, so they must be approximated in practice. We will do so in the next section. Here, we assess the power of our techniques from a principled perspective, ignoring this source of complication. We consider what would happen if we did use the exact label sets as defined.

Theoretical Results with Exact Labels

Catching globally relevant labels matches full bisimulation in that it yields a perfect heuristic (cf. Theorem 1); greedy bisimulation does not give that guarantee. Compared to both full bisimulation and greedy bisimulation, catching globally relevant labels is potentially better because it makes less distinctions. This can yield an exponential advantage:

Proposition 1 *There exist families of planning tasks $\{\Pi_n\}$, with variable subsets $\{V_n\}$ and globally relevant labels $\{G_n\}$, so that M&S abstractions over V_n are exponentially smaller with the shrinking strategy using G_n-catching bisimulation, than with the shrinking strategies using either of bisimulation or greedy bisimulation.*

Our example showing this introduces exponentially many distinctions in (greedy) bisimulation by operators that, although they can be used to construct a solution, are not used in any optimal solution and are thus not globally relevant.

Likewise, imposing a radius on the caught labels can have an exponential advantage (while still guaranteeing A* to be efficient, cf. Theorem 2):

Proposition 2 *There exist families of planning tasks $\{\Pi_n\}$, with variable subsets $\{V_n\}$, globally relevant labels $\{G_n\}$, and $h^*(s_0)$-relevant labels $\{R_n\}$, so that M&S abstractions over V_n are exponentially smaller with the shrinking strategy using R_n-catching bisimulation, than with the shrinking strategies using either of G_n-catching bisimulation, bisimulation, or greedy bisimulation.*

This situation can arise from operators that do participate in optimal solutions, but only within an irrelevant region of the state space (reached by making a bad action choice).

Propositions 1 and 2 hold regardless whether or not Nissim et al.'s label reduction technique is used. Note that the situations underlying the proofs are quite natural. In particular, as we will see in the next sub-section, most IPC benchmarks contain at least some operators as described. This

notwithstanding, to our knowledge none of the benchmarks actually contains a family as claimed in Propositions 1 and 2. Intuitively, the described situations do occur, but to a lesser extent. An exception is Dining-Philosophers in a direct finite-domain planning encoding, for which Nissim et al. showed that greedy bisimulation yields perfect heuristics with polynomial effort. The same is true when catching globally or $h^*(s_0)$-relevant labels.

Empirical Results with Exact Labels

We ran M&S with no shrinking and no reachability pruning (no removal of non-reachable abstract states during M&S) to compute the full state space, and thus the exact label sets; Table 1 shows results on the 172 IPC benchmark instances where this process did not run out of memory. We show, summed-up per instance, the label set size and the size of the largest abstractions generated during M&S, when catching all labels ("All") vs. the globally relevant labels ("Global") vs. the $h^*(s_0)$-relevant labels ("$h^*(s_0)$").

	Σ number of labels			Σ maximal abstraction size		
domain	All	Global	$h^*(s_0)$	All	Global	$h^*(s_0)$
blocks	462	459	**453**	5338545	5338437	**5337835**
depots	72	**48**	**48**	26928	**12402**	**12402**
driverlog	448	**383**	**383**	1046925	1046925	1046925
gripper	232	**176**	**176**	712	712	712
logistics00	672	366	**364**	1314376	1314376	1314376
logistics98	278	**173**	**173**	4157536	4157536	4157536
miconic	5700	4070	**4069**	1314030	1314660	1314660
mystery	154	126	**94**	41408	39600	**33768**
nomystery11	5198	**4501**	**4501**	9688	**8464**	**8464**
openstack08	400	**383**	**383**	21396	21396	21396
openstack11	575	**515**	**515**	9048	9048	9048
parcprint08	158	115	**103**	**359**	374	392
parcprint11	59	**39**	**39**	**241**	257	257
pathways	61	**30**	**30**	97	97	97
pegsol08	166	166	**128**	180720	180720	**94305**
psr	1993	1753	**1745**	106780	**103596**	**103596**
rovers	161	**100**	**100**	8886	**1920**	**1920**
satellite	456	**326**	**326**	11302	**8488**	**8488**
scanaly08	2724	**1224**	**1224**	40320	40320	40320
scanaly11	1168	**668**	**668**	20192	20192	20192
tpp	38	38	38	276	276	276
transport08	1400	1232	**1192**	279850	**279733**	280883
transport11	424	**400**	**400**	160000	160000	160000
trucks	597	**203**	**203**	**8175**	8423	8423
zeno	2246	1581	**1512**	4689384	4689384	**4689056**
Σ	26112	19345	**19137**	18787174	18757336	**18665327**

Table 1: Summed-up sizes of exact label sets (all vs. globally relevant vs. $h^*(s_0)$-relevant), and of maximum abstraction sizes during M&S for bisimulation catching these.

A quick look at the left-hand side of the table confirms that there tend to be quite some labels that can be ignored without sacrificing heuristic quality. The single domain with no irrelevant labels at all is TPP. Often, only about two thirds of the labels are $h^*(s_0)$-relevant; in Trucks, only one third are. At the same time, a look at the right-hand side of the table shows that the reduced label sets are not very effective in reducing abstraction size. In only 10 out of 24 domains with reduced labels, the maximal abstraction size is reduced as well. The reduction is typically small, except in a few domains like PegSol (factor 1.92) and Rovers (factor 4.63). In two cases (ParcPrinter and Trucks), the size actually grows.[5]

The present data should be treated with caution as the instances considered are very small; the abstraction size reductions might be more significant in larger instances. This notwithstanding, in practice it may be advisable to approximate the label subsets aggressively, catching less labels in the hope to reduce abstraction size more, while not losing too much information. We consider such methods next.

Results Using Approximate Label Sets

We describe our label-subset approximation techniques, then run experiments on the standard IPC benchmarks.

Approximation Techniques

The word "relevant" in the names of the label sets identified in Definitions 4 and 5 was chosen because the intuition behind these – subsets of operators used in optimal plans – is very close to previous notions of relevance (e. g., (Nebel, Dimopoulos, and Koehler 1997; Brafman 2001; Hoffmann and Nebel 2001)). This creates a potential for synergy. We implemented one method inspired by this, and one method that integrates particularly well with M&S:

- **Backward h^1.** This is a variant of backward-chaining relevance detection, using a straightforward backwards version of the equations defining h^1 (Haslum and Geffner 2000). We collect all operators that appear within the radius R given by the product of (forward) $h^1(s_0)$ and a parameter $\beta \in [0, 1]$. Note that, for h^m with large m, the selected labels would be exactly the $h^*(s_0)$-relevant ones. Setting β allows to select less labels, controlling the trade-off between abstraction size and accuracy. For $\beta = 0$, we use the smallest β yielding a non-empty label set.
- **Intermediate Abstraction (IntAbs).** We run full bisimulation until abstraction size has reached a parameter M. The labels are then collected by applying either of Definition 4 or 5 to the present abstraction, and M&S continues with bisimulation catching that label subset. With very large M (and when not removing non-reachable abstract states), the label set would be exact. Small M results in smaller labels sets because non-0 cost operators on variables not yet merged will not be considered relevant.

Neither technique guarantees, in general, to catch *all* globally relevant/$h^*(s_0)$-relevant labels. They are practical approximations whose merits we now evaluate experimentally.

Experiments

Our techniques are implemented in Fast Downward, and all results we report use the same A* implementation. We ran a total of 32 M&S configurations, plus two competing heuristics, on 1396 instances from 44 IPC benchmark domain suites. To make these 47464 runs feasible, the runtime for each was limited to 5 minutes. The memory limit was

[5]The discrepancy with Lemma 5 is due to removal of non-reachable abstract states, done in our code, but not in the lemma. In rare cases, the coarser abstraction (catching less labels) may produce more reachable abstract states when merged with another variable. Our TR contains an example illustrating this.

Approach	IntAbs Global				Backward h^1								strict-greedy bisimulation				Nissim et al.				BJOLP	LM-cut
N	10K	100K	∞	∞	10K			∞					10K	100K	∞	∞	10K	100K	∞	∞		
M/β/Nissim et al. variant	10K	10K	10K	100K	0.25	0.5	1	0	0.25	0.5	0.75	1	10K	10K	10K	100K	full	s-greedy	full	s-greedy		
airport	*22	*22	*22	*22	19	19	19	*22	3	1	1	1	*22	*22	*22	*22	19	16	1	*22	**28**	25
barman-opt11-strips	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	0	0	0	0	**4**	**4**	**4**	**4**	**4**	**4**	0	**4**	**4**	0
blocks	*21	*21	*21	18	*21	*21	*21	18	14	9	9	9	*21	*21	*21	*21	*21	*21	9	*21	26	**28**
depot	**7**	**7**	**7**	6	6	6	6	6	6	1	1	1	**7**	**7**	**7**	**7**	6	**7**	1	**7**	**7**	**7**
driverlog	12	12	12	12	12	12	12	*13	*13	6	5	5	12	12	12	12	12	12	5	12	**14**	13
elevators-opt11-strips	9	9	10	*12	9	9	9	9	0	0	0	0	9	9	9	9	9	9	0	9	12	**15**
floortile-opt11-strips	2	3	3	**7**	2	3	3	2	3	6	6	6	2	2	2	2	3	3	6	2	2	6
freecell	*15	*15	*15	6	*15	*15	*15	*15	13	6	2	1	*15	*15	*15	*15	*15	7	1	*15	**53**	9
grid	1	1	1	0	**2**	**2**	**2**	**2**	**2**	0	0	0	**2**	**2**	**2**	**2**	**2**	1	0	**2**	**2**	1
gripper	7	10	10	**20**	7	7	11	7	7	7	7	**20**	7	7	7	7	11	7	**20**	7	7	6
logistics00	16	16	16	16	18	**20**	**20**	18	18	10	10	10	16	16	16	16	**20**	16	10	16	**20**	**20**
logistics98	4	4	4	*5	4	4	4	*5	*5	3	2	2	4	4	4	4	4	4	2	4	**6**	**6**
miconic	51	52	52	55	51	51	*57	51	51	51	55	50	50	50	50	50	*57	55	51	50	**141**	140
mprime	22	22	22	13	**23**	**23**	19	22	22	16	2	1	22	22	22	22	19	10	1	22	20	20
mystery	**15**	**15**	**15**	14	**15**	14	13	13	13	11	4	3	**15**	**15**	**15**	**15**	13	10	3	**15**	**15**	**15**
nomystery-opt11-strips	12	15	15	**19**	16	18	18	16	16	9	12	12	12	12	12	12	18	15	12	12	18	13
openstacks-opt11-strips	**14**	**14**	**14**	12	**14**	**14**	**14**	**14**	**14**	**14**	**14**	1	**14**	**14**	**14**	**14**	**14**	**14**	1	**14**	10	11
openstacks-strips	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**
parcprinter-opt11-strips	*12	*12	*12	*12	11	11	*12	11	9	9	8	8	11	11	11	11	*12	*12	8	11	9	**13**
parking-opt11-strips	**5**	4	4	0	2	2	3	0	0	0	0	0	**5**	**5**	**5**	**5**	3	0	0	**5**	1	1
pathways-noneg	*4	*4	*4	*4	*4	*4	*4	*4	*4	3	*4	*4	*4	*4	*4	*4	*4	*4	*4	*4	4	**5**
pegsol-opt11-strips	**19**	**19**	**19**	**19**	17	17	18	17	17	8	8	0	17	17	17	17	**19**	18	0	17	17	17
pipesworld-notankage	15	15	15	15	*16	15	15	3	3	2	1	2	15	15	15	15	15	11	2	15	**17**	15
pipesworld-tankage	12	12	12	10	14	14	14	2	2	2	1	2	**16**	**16**	**16**	15	14	13	2	**16**	11	8
psr-small	49	49	49	49	49	49	49	49	49	49	49	44	49	**50**	**50**	**50**	49	**50**	44	**50**	49	48
rovers	6	6	6	6	6	6	6	6	6	6	4	4	6	6	6	6	6	**7**	4	6	**7**	**7**
satellite	6	6	6	6	7	7	6	**8**	**8**	**8**	6	6	6	6	6	6	6	6	6	6	7	7
scanalyzer-opt11-strips	**10**	**10**	**10**	8	9	9	9	3	3	3	3	3	**10**	**10**	**10**	6	9	9	3	3	3	**10**
sokoban-opt11-strips	19	19	19	18	19	19	19	16	11	5	3	1	19	19	19	19	19	**20**	0	19	18	19
tidybot-opt11-strips	12	11	11	0	8	1	1	**14**	4	1	1	1	13	12	12	12	4	0	0	12	**14**	11
tpp	6	6	6	**7**	6	6	6	6	6	5	5	5	6	6	6	6	6	**7**	5	6	6	6
transport-opt11-strips	6	6	6	**8**	6	6	6	6	1	1	1	1	6	6	6	6	6	6	1	6	6	6
trucks-strips	5	5	5	5	6	*7	5	6	6	6	4	4	5	5	5	5	6	6	4	5	7	**9**
visitall-opt11-strips	9	9	9	10	9	9	9	8	8	8	8	8	**12**	**12**	**12**	**12**	9	9	8	**12**	9	10
woodworking-opt11-strips	6	6	6	7	4	6	6	5	2	1	1	1	7	*9	*9	*9	6	*9	2	*9	7	**10**
zenotravel	9	9	9	11	**12**	**12**	11	**12**	**12**	**12**	7	7	9	9	9	9	11	9	7	9	10	11
Σ	585	591	*593	575	578	579	585	538	449	358	320	270	588	*593	*593	584	591	547	270	579	**715**	698
Σ w/o miconic & freecell	519	524	526	514	512	513	513	472	385	301	263	219	523	*528	*528	519	519	485	218	514	521	**549**
Σ M&S built	1383	1341	1336	1049	1236	1196	1178	989	687	501	352	270	*1385	1357	1347	1290	1174	1018	270	1264		

Table 2: Selected coverage data in IPC benchmarks. Best results overall (of all M&S heuristics) are highlighted in bold (with a "*"). "Σ M&S built": number of tasks for which computing the M&S abstraction did not exceed the available time/memory.

2 GB. The runs were conducted on machines equipped with two quad-core CPUs (AMD Opteron 2384). Coverage data is shown in Table 2. To save space, we omit domains from IPC'08 that were run also in IPC'11.

We run BJOLP (Domshlak et al. 2011) and LM-cut (Helmert and Domshlak 2009) because they were the two non-M&S components in Fast Downward Stone Soup, the portfolio winning the 1st prize in the track for optimal planners at IPC'11. We ran 9 M&S configurations from the work by Nissim et al., setting $N \in \{10K, 100K, \infty\}$ and using either full bisimulation, or greedy bisimulation, or **strict-greedy bisimulation** (s-greedy). The latter is the variant of Definition 2 catching all transitions $(s, l, s') \in T$ where $h^*(s') < h^*(s)$, rather than $h^*(s') \leq h^*(s)$. This variant is not mentioned by Nissim et al., but actually is what is run in their experiments and in the IPC. As for the parameter N, in all M&S variants, this is a bound on abstraction size reaching which forces the shrinking strategy to aggregate more states, dropping any bisimulation guarantees (Helmert, Haslum, and Hoffmann 2007; Nissim, Hoffmann, and Helmert 2011a). For $N = \infty$, the bisimulation guarantee is always held up (and the abstraction might run out of memory). Given the limited space in Table 2, we show data for 4 of the 9 Nissim et al. configurations: $N = 10K$ with full bisimulation, the one with highest overall coverage; $N = \infty$ with full bisimulation, for reference (in difference to all other M&S configurations here, this guarantees the heuristic to be perfect); and the two configurations taking part in Fast Downward Stone Soup, $N = 100K$ and $N = \infty$ with strict-greedy bisimulation.[6]

We also run 4 new M&S variants using strict-greedy bisimulation, with the parameter M of our Intermediate Abstraction (IntAbs) label approximation. These configurations start with full bisimulation, then switch to s-greedy bisimulation once abstraction size M is reached. This allows for a very direct comparison with our IntAbs configurations: the only difference to these lies in their use of label-catching bisimulation, rather than s-greedy bisimulation, after M is reached. We do not show data for IntAbs with Definition 5 ($h^*(s_0)$-relevant labels), because these configurations are dominated by the ones using Definition 4 (globally relevant labels). Compared to the variants orginally designed by Nissim et al, the new s-greedy variants have a significant adavantage in total coverage. They also have a small such advantage vs. the IntAbs variants. However, the former have the edge in a larger number of individual domains. The respective configuration with best coverage is strictly better for IntAbs in 15 domains, is equally good in 23 domains, and is worse only in 6 domains. An interesting observation within IntAbs is that, as expected, smaller M yields more greedy abstractions. For $N = \infty$, $M = 10K$ completes 1336 abstractions, vs. 1049 completed by $M = 100K$.

We finally run 11 M&S variants with the Backward h^1 label-catching strategy: $N = 10K$ with 4 values of β ($\beta = 0.75$ not shown because it is always dominated by one of the others); and $N = \infty$ with 7 values of β. For

[6]Actually, $N = 200K$ was used in the IPC; the performance for $N = 100K$ is almost identical to that.

$\|P\|$	2	4				6				2			4			6		
Design	BL	FDSS	BL+O	BL+N	BL+ON	FDSS+N	BL+O	BL+N	BL+ON	O	N	ON	O	N	ON	O	N	ON
$\|C\|$	0	0	13	13	26	13	13	13	26	13	13	26	13	13	26	13	13	26
Upper bound	770	805	825	823	833	830	825	823	833	658	649	673	658	649	673	658	649	673
Best P	770	805	825	819	827	830	825	823	833	656	630	656	658	647	671	658	649	673

Table 3: Portfolios. "$|P|$": number of components within portfolio. "Design": portfolio design space (see text). "$|C|$": number of components to choose from. "Upper bound": solved by any possible component. "Best P": best coverage of any portfolio.

$N = 10K$, β has hardly any effect since enforcing the bound makes the abstraction very greedy anyhow. By contrast, for $N = \infty$, smaller β decreases computational effort drastically (consider the bottom row in Table 2). In effect, in 35 of the 44 domains, coverage increases monotonically as we decrease β. Note also that, for $\beta = 1.0$, performance is almost identical to that of full bisimulation with $N = \infty$. Indeed, the number of labels caught (not shown here) is typically close to the total number of labels.

Comparing the per-domain perfomance of the Backward h^1 configurations with the IntAbs configurations, the latter have a slight edge. The configuration with best coverage is strictly better for Backward h^1 in 11 domains, is equally good in 18 domains, and is worse in 15. Comparing the new M&S variants (IntAbs and Backward h^1) with all "old" ones (including the novel s-greedy variants), the best-coverage configuration is better for new M&S in 10 domains, equally good in 23, and worse in 11. Comparing the new M&S variants against all other planners, the best-coverage configuration is better for new M&S in 5 domains, equally good in 16, and worse in 23. Altogether, the new heuristics are certainly not a breakthrough in coverage of cost-optimal planners, but they can contribute. We reconfirm this below by considering portfolios built from different subsets of configurations.

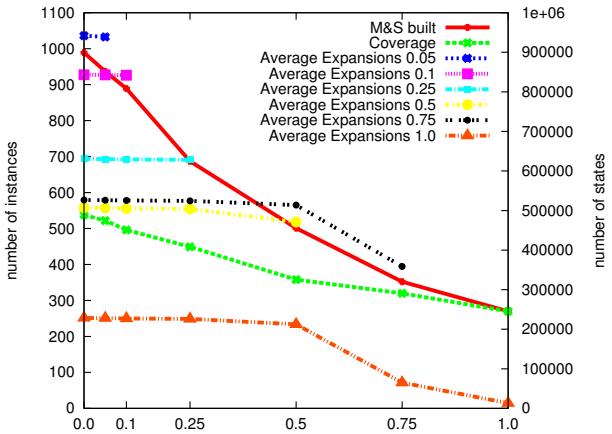

Figure 1: Scaling β in Backward h^1 with $N = \infty$.

Figure 1 examines more closely how β trades off abstraction effort against accuracy. The coverage and "M&S built" data (left y-axis) are as in Table 2. "Expansions X" (right y-axis) shows the average number of expanded states in the subset of instances solved by all configurations where $\beta \leq X$. That subset contains much larger instances for smaller β, hence the average expansions grow. Note however that there is a consistent pattern *within* each of these curves. Expansions increase a lot as we step from $\beta = 0.75$ to $\beta = 0.5$ (e. g., from 64976 to 212362 for "Expansions 1.0"), but remain almost constant at both sides of this step. This suggests a kind of phase transition, where for $\beta \geq 0.75$ the heuristic is close to perfect, whereas for β going below 0.5 it is quite bad, and does not get a lot worse while still dramatically reducing abstraction effort. The latter does a lot to help coverage, and one could try to catch even less labels when $\beta = 0$. One could also try to add complementary label selection techniques, in the hope to push the "phase transition" to smaller β. Both are topics for future work.

Different M&S heuristics often have complementary strengths. Table 3 examines this in detail, listing the best performance any sequential portfolio of a given size $|P| \in \{2, 4, 6\}$ can obtain, when selecting its components from particular subsets of configurations. Comparisons should be made only within groups of portfolios with same $|P|$, as each component uses 5 minutes and thus $|P|$ determines the computational resources used. In the "Design" row, "BL" is BJOLP+LM-cut, and "FDSS" has the same configurations as Fast Downward Stone Soup (cf. above). By "X+Y" we denote portfolios P in which the components X are fixed and only the remaining $|P| - |X|$ components are selected from Y. "O" ("Old") refers to the 13 "old" M&S configurations we run here. "N" ("New") refers to 13 of the IntAbs and Backward h^1 configurations (to obtain groups "O" and "N" of same size, we omitted Backward h^1 with $N = \infty$ and $\beta > 0$). "BL" is included only for reference. The data for $|P| = 4$ and "BL+Y" design shows that, in our setting here, different M&S variants than in "FDSS" yield better coverage; the data for $|P| = 6$ and "BL+Y" design shows that adding even more M&S configurations still improves the outcome. Generally, portfolios of only "O" M&S configurations are better than those of only "N" ones, but the best option is to combine the two.

Conclusion

Label-catching bisimulation is very appealing in principle: it is invariant over M&S, guarantees a perfect heuristic if we catch all relevant labels, may be exponentially smaller than full bisimulation even in this case, and allows a fine-grained effort/accuracy trade-off by plugging in approximations of relevance. At the same time, our empirical results are a bit disappointing, performance being improved only in few domains. As indicated, one could try to design different relevance approximations. The authors' speculation is that there is more potential in *combining* M&S heuristics, i. e., automatically constructing sets of heuristics specifically designed to be complementary, for a given planning task.

Acknowledgments. Work performed while Michael Katz and Jörg Hoffmann were employed by INRIA, Nancy, France. Michael Katz was supported by the French National Research Agency (ANR), project ANR-10-CEXC-003-01.

References

Brafman, R. 2001. On reachability, relevance, and resolution in the planning as satisfiability approach. *Journal of Artificial Intelligence Research* 14:1–28.

Domshlak, C.; Helmert, M.; Karpas, E.; Keyder, E.; Richter, S.; Röger, G.; Seipp, J.; and Westphal, M. 2011. BJOLP: The big joint optimal landmarks planner. IPC 2011 planner abstracts.

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

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In Chien, S.; Kambhampati, S.; and Knoblock, C. A., eds., *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2000)*, 140–149. AAAI Press.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 162–169. AAAI Press.

Helmert, M.; Röger, G.; Seipp, J.; Karpas, E.; Hoffmann, J.; Keyder, E.; Nissim, R.; Richter, S.; and Westphal, M. 2011. Fast Downward Stone Soup. IPC 2011 planner abstracts.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In Boddy, M.; Fox, M.; and Thiébaux, S., eds., *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS 2007)*, 176–183. AAAI Press.

Hoffmann, J., and Nebel, B. 2001. RIFO revisited: Detecting relaxed irrelevance. In Cesta, A., and Borrajo, D., eds., *Pre-proceedings of the Sixth European Conference on Planning (ECP 2001)*, 325–336.

Katz, M.; Hoffmann, J.; and Helmert, M. 2012. How to relax a bisimulation? Technical Report 7901, INRIA. Available at http://hal.inria.fr/hal-00677299.

Milner, R. 1990. Operational and algebraic semantics of concurrent processes. In van Leeuwen, J., ed., *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics*. Elsevier and MIT Press. 1201–1242.

Nebel, B.; Dimopoulos, Y.; and Koehler, J. 1997. Ignoring irrelevant facts and operators in plan generation. In Steel, S., and Alami, R., eds., *Recent Advances in AI Planning. 4th European Conference on Planning (ECP 1997)*, volume 1348 of *Lecture Notes in Artificial Intelligence*, 338–350. Springer-Verlag.

Nissim, R.; Hoffmann, J.; and Helmert, M. 2011a. Computing perfect heuristics in polynomial time: On bisimulation and merge-and-shrink abstraction in optimal planning. In Walsh, T., ed., *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*, 1983–1990. AAAI Press/IJCAI.

Nissim, R.; Hoffmann, J.; and Helmert, M. 2011b. The Merge-and-Shrink planner: Bisimulation-based abstraction for optimal planning. IPC 2011 planner abstracts.