Red-Black Relaxed Plan Heuristics Reloaded

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

Abstract

Despite its success, the delete relaxation has significant pitfalls. In an attempt to overcome these pitfalls, recent work has devised so-called *red-black relaxed plan heuristics*, where red variables take the relaxed semantics (accumulating their values), while black variables take the regular semantics. These heuristics were shown to significantly improve over standard delete relaxation heuristics. However, the experiments also brought to light a major weakness: Being based on repairing fully delete-relaxed plans, the returned estimates depend on arbitrary choices made in such plans. This can lead to huge over-estimation in arbitrary subsets of states. Here we devise a new red-black planning method not based on repairing relaxed plans, getting rid of much of this variance. Our experiments show a significant improvement over previous red-black relaxed plan heuristics, and other related methods.

Introduction

The *delete relaxation*, that we will also refer to as the *monotonic relaxation* here, has played a key role in advancing planning systems over the last decade. It was particularly successful in satisficing planning (not giving an optimality guarantee), which we focus on here. In the monotonic relaxation, state variables accumulate their values, rather than switching between them. The generation of (non-optimal) plans in monotonic planning is polynomial-time (Bylander 1994), allowing for the use of such plans for the generation of (non-admissible) heuristic functions. Such so-called *relaxed plan heuristics* have been of paramount importance for the success of satisficing planning during the last decade, e. g., for the HSP, FF, and LAMA planning systems (Bonet and Geffner 2001; Hoffmann and Nebel 2001; Richter and Westphal 2010).

This success notwithstanding, the delete relaxation has significant pitfalls, e. g., in planning with non-replenishable resources, whose consumption is completely ignored within the relaxation. Recent years thus have seen active research aiming at taking *some* deletes into account, e. g. (Fox and Long 2001; Gerevini, Saetti, and Serina 2003; Helmert 2004; Helmert and Geffner 2008; Keyder and Geffner 2008; Baier and Botea 2009; Cai, Hoffmann, and Helmert 2009; Haslum 2012; Keyder, Hoffmann, and Haslum 2012; Katz, Hoffmann, and Domshlak 2013b). In particular, Katz et al. (2013b) introduced *red-black planning*, in which a subset of "red" state variables takes on the monotonic, value-accumulating semantics, while the other "black" variables retain the regular semantics.

Building on that theoretical work, Katz et al. (2013a) devise red-black relaxed plan heuristics and demonstrate their potential. Despite impressive empirical performance in some domains, as Katz et al. point out, the technique's behavior is not reliable. In many domains, the effect of the new heuristics on search exhibits a huge variance, decreasing the search space by a factor of 100 in some instances, and increasing it by a similar factor in others. Katz et al. conjecture that this is because their red-black planning methods work by repairing relaxed plans, and are thus too dependent on arbitrary choices made in such plans. This can lead to huge over-estimation in some search states but not in others. Greedy best-first search is bound to be brittle with respect to such variance, which can explain said lack of reliability.

We introduce a new red-black planning algorithm not based on repairing relaxed plans, getting rid of much of this issue. Instead of following the relaxed plan's actions and action ordering, we merely follow the set of *red facts the relaxed plan uses*: The preconditions and goals on red variables in the relaxed plan. We allow arbitrary action choices to achieve these. Our experiments show that this typically (sometimes dramatically) reduces over-estimation, and that significant performance improvements are obtained over the previous method (and over the baseline FF heuristic), in most domains. Furthermore, the quality of red-black plan heuristics depends also on the choice of black variables. We devise improvements to the techniques suggested by Katz et al., and we explore a method to interpolate between these.

Like Katz et al. (2013a), we rely on a tractable fragment of red-black planning identified by an *acylic black causal graph* (the projection of the causal graph onto the black variables), and by requiring all black variables to be invertible in a particular sense (*RSE-invertible*). Also like Katz et al., in our experiments we focus on a simpler fragment where the black causal graph does not contain any arcs at all.

The paper is organized as follows. We start by giving the background. We then summarize the previous red-black planning method, and detail its over-estimation issues in some examples. We present our new method, prove its cor-

rectness, and discuss its effect in these same examples. We present our experiments and conclude.

Background

A **finite-domain representation (FDR)** planning task is a quadruple $\Pi = \langle V, A, I, G\rangle$. V is a set of *state variables*, where each $v \in V$ is associated with a finite domain $\mathcal{D}(v)$. A complete assignment to V is called a *state*. I is the *initial state*, and the *goal* G is a partial assignment to V. A is a finite set of *actions*. Each action a is a pair $\langle \mathsf{pre}(a), \mathsf{eff}(a)\rangle$ of partial assignments to V called *precondition* and *effect*, respectively. We sometimes refer to (partial) assignments as sets of *facts*, i. e., variable-value pairs (v, d).

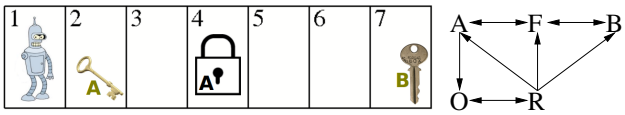
The semantics of FDR tasks is as follows. For a partial assignment p, $\mathcal{V}(p) \subseteq V$ denotes the subset of state variables instantiated by p. For $V' \subseteq \mathcal{V}(p)$, $p[V']$ denotes the value of V' in p. Action a is applicable in state s iff $s[\mathcal{V}(\mathsf{pre}(a))] = \mathsf{pre}(a)$, i. e., iff $s[v] = \mathsf{pre}(a)[v]$ for all $v \in \mathcal{V}(\mathsf{pre}(a))$. Applying a in s changes the value of $v \in \mathcal{V}(\mathsf{eff}(a))$ to $\mathsf{eff}(a)[v]$; the resulting state is denoted by $s[\![a]\!]$. By $s[\![\langle a_1, \ldots, a_k\rangle]\!]$ we denote the state obtained from sequential application of $a_1, \ldots, a_k$ starting at s. An action sequence is a *plan* if $I[\![\langle a_1, \ldots, a_k\rangle]\!][\mathcal{V}(G)] = G$.

Figure 1 (a) shows the illustrative example given by Katz et al. (2013b), that we also adopt here. The example is akin to the GRID benchmark, and is encoded in FDR using the following state variables: R, the robot position in $\{1, \ldots, 7\}$; A, the key A position in $\{R, 1, \ldots, 7\}$; B, the key B position in $\{R, 1, \ldots, 7\}$; F in $\{0, 1\}$ saying whether the robot hand is free; O in $\{0, 1\}$ saying whether the lock is open. We can *move* the robot from i to $i + 1$, or vice versa, if the lock is open or $\{i, i + 1\} \cap \{4\} = \emptyset$. We can *take* a key if the hand is free, *drop* a key we are holding, or *open* the lock if the robot is at 3 or 5 and holds key A. The goal requires key B to be at 1. An optimal plan moves to 2, takes key A, moves to 3, opens the lock, moves to 7, drops key A and takes key B, moves back to 1 and drops key B.

A **monotonic finite-domain representation (MFDR)** planning task is a quadruple $\Pi = \langle V, A, I, G\rangle$ exactly as for FDR tasks, but the semantics is different. An MFDR state s is a function that assigns each $v \in V$ a non-empty subset $s[v] \subseteq \mathcal{D}(v)$ of its domain. An MFDR action a is applicable in state s iff $\mathsf{pre}(a)[v] \in s[v]$ for all $v \in \mathcal{V}(\mathsf{pre}(a))$, and applying it in s changes the value of $v \in \mathcal{V}(\mathsf{eff}(a))$ to $s[v] \cup \{\mathsf{eff}(a)[v]\}$. An action sequence $\langle a_1, \ldots, a_k\rangle$ is a *plan* if $G[v] \in I[\![\langle a_1, \ldots, a_k\rangle]\!][v]$ for all $v \in \mathcal{V}(G)$.

Plans for MFDR tasks can be generated in polynomial time (this follows directly from Bylander's (1994) results). A key part of many planning systems is based on exploiting this property for deriving heuristic estimates, via the notion of monotonic, or delete, relaxation. The **monotonic relaxation** of an FDR task $\Pi = \langle V, A, I, G\rangle$ is the MFDR task $\Pi^+ = \Pi$. The **optimal delete relaxation heuristic** $h^+(\Pi)$ is the length of a shortest possible plan for Π^+. For arbitrary states s, $h^+(s)$ is defined via the MFDR task $\langle V, A, s, G\rangle$. If π^+ is a plan for Π^+, then π^+ is a **relaxed plan** for Π.

A relaxed plan for the example takes key A, opens the lock, moves to 7, takes key B (without first dropping key A), and drops key B at 1 (without first moving back there). We get $h^+(\Pi) = 10$ whereas the real plan needs 17 steps.



(a) (b)

Figure 1: An example (a), and its causal graph (b).

We will use two standard structures to identify special cases of planning. The **causal graph** CG_Π of a task Π is a digraph with vertices V. An arc (v, v') is in CG_Π iff $v \neq v'$ and there exists an action $a \in A$ such that $(v, v') \in [\mathcal{V}(\mathsf{eff}(a)) \cup \mathcal{V}(\mathsf{pre}(a))] \times \mathcal{V}(\mathsf{eff}(a))$. The **domain transition graph** $DTG_\Pi(v)$ of a variable $v \in V$ is a labeled digraph with vertices $\mathcal{D}(v)$. The graph has an arc (d, d') induced by action a iff $\mathsf{eff}(a)[v] = d'$, and either $\mathsf{pre}(a)[v] = d$ or $v \notin \mathcal{V}(\mathsf{pre}(a))$. The arc is labeled with its **outside condition** $\mathsf{pre}(a)[V \setminus \{v\}]$ and its **outside effect** $\mathsf{eff}(a)[V \setminus \{v\}]$.

Consider again the example. Figure 1 (b) shows the causal graph: R is a prerequisite for changing every other variable. Each key is interdependent with F because taking/dropping them affects both. Key A influences O, which influences R. $DTG_\Pi(R)$ has arcs $(i, i + 1)$ and $(i + 1, i)$, all with empty outside effect, and with empty outside condition except if $\{i, i + 1\} \cap \{4\} \neq \emptyset$ in which case the outside condition is $\{(O, 1)\}$. $DTG_\Pi(F)$ has an arc $(1, 0)$ for every *take*(x, y) action where $x \in \{1, \ldots, 7\}$ and $y \in \{A, B\}$, with outside condition $\{(R, x), (y, x)\}$ and outside effect $\{(y, R)\}$, as well as an arc $(0, 1)$ for every *drop*(x, y) action, with outside condition $\{(R, x), (y, R)\}$ and outside effect $\{(y, x)\}$.

Katz et al. (2013b) view FDR and MFDR as cases in which *all* state variables adopt value-switching and value-accumulating semantics, respectively. They interpolate between these extremes by what they call *red-black planning*.

A **red-black (RB)** planning task is a tuple $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G\rangle$ where V^{B} is a set of *black state variables*, V^{R} is a set of *red state variables*, and everything else is exactly as for FDR and MFDR tasks. A state s assigns each $v \in V^{\mathsf{B}} \cup V^{\mathsf{R}}$ a non-empty subset $s[v] \subseteq \mathcal{D}(v)$, where $|s[v]| = 1$ for all $v \in V^{\mathsf{B}}$. An RB action a is applicable in state s iff $\mathsf{pre}(a)[v] \in s[v]$ for all $v \in \mathcal{V}(\mathsf{pre}(a))$. Applying a in s changes the value of $v \in \mathcal{V}(\mathsf{eff}(a)) \cap V^{\mathsf{B}}$ to $\{\mathsf{eff}(a)[v]\}$, and changes the value of $v \in \mathcal{V}(\mathsf{eff}(a)) \cap V^{\mathsf{R}}$ to $s[v] \cup \{\mathsf{eff}(a)[v]\}$. An action sequence $\langle a_1, \ldots, a_k\rangle$ is a *plan* if $G[v] \in I[\![\langle a_1, \ldots, a_k\rangle]\!][v]$ for all $v \in \mathcal{V}(G)$.

In the example, if variables R, A, B, O are red and F is black, then (in difference to the relaxed plan) the robot needs to drop key A before taking key B. If R is black as well, then the robot needs to move back to 1 before dropping key B, rendering the red-black plan a real plan.

RB obviously generalizes both FDR and MFDR. Given an FDR planning task $\Pi = \langle V, A, I, G\rangle$ and a subset $V^{\mathsf{R}} \subseteq V$ of its variables, the **red-black relaxation** of Π relative to V^{R} is the RB task $\Pi^{*+}_{V^{\mathsf{R}}} = \langle V \setminus V^{\mathsf{R}}, V^{\mathsf{R}}, A, I, G\rangle$. A plan for $\Pi^{*+}_{V^{\mathsf{R}}}$ is a **red-black relaxed plan** for Π, and the length of a shortest possible red-black relaxed plan is denoted $h^{*+}_{V^{\mathsf{R}}}(\Pi)$. For arbitrary states s, $h^{*+}_{V^{\mathsf{R}}}(s)$ is defined via the RB task $\langle V \setminus V^{\mathsf{R}}, V^{\mathsf{R}}, A, s, G\rangle$. It is easy to see that $h^{*+}_{V^{\mathsf{R}}}$ is consis-

Algorithm : UNRELAX(Π, π^+)

main

// $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G\rangle$ *and* $\pi^+ = \langle a_1, \ldots, a_n\rangle$

$\pi \leftarrow \langle a_1\rangle$

for $i = 2$ **to** n

do
- **if** $\mathsf{pre}(a_i)[V^{\mathsf{B}}] \not\subseteq I[\![\pi]\!]$
  - **then** $\pi \leftarrow \pi \circ$ ACHIEVE$(\pi, \mathsf{pre}(a_i)[V^{\mathsf{B}}])$
- $\pi \leftarrow \pi \circ \langle a_i\rangle$

if $G[V^{\mathsf{B}}] \not\subseteq I[\![\pi]\!]$

then $\pi \leftarrow \pi \circ$ ACHIEVE$(\pi, G[V^{\mathsf{B}}])$

procedure ACHIEVE(π, g)

$F \leftarrow I \cup \bigcup_{a\in\pi} \mathsf{eff}(a)$

$A^{\mathsf{B}} \leftarrow \{a^{\mathsf{B}} \mid a \in A, a^{\mathsf{B}} = \langle \mathsf{pre}(a)[V^{\mathsf{B}}], \mathsf{eff}(a)[V^{\mathsf{B}}]\rangle,$
$\mathsf{pre}(a) \subseteq F, \mathsf{eff}(a)[V^{\mathsf{B}}] \subseteq F\}$

$\langle a'^{\mathsf{B}}_1, \ldots, a'^{\mathsf{B}}_k\rangle \leftarrow$ an FDR plan for $\Pi^{\mathsf{B}} = \langle V^{\mathsf{B}}, A^{\mathsf{B}}, I[\![\pi]\!][V^{\mathsf{B}}], g\rangle$

return $\langle a'_1, \ldots, a'_k\rangle$

Figure 2: Algorithm presented by Katz et al. (2013a).

tent and dominates h^+, and if $V^{\mathsf{R}} = \emptyset$ then $h^{*+}_{V^{\mathsf{R}}}$ is perfect. Computing $h^{*+}_{V^{\mathsf{R}}}$ is hard, and Katz et al. propose to use upper-approximation by satisficing red-black planning, in analogy to the successful strategies for relaxed planning. For this to be practical, satisficing red-black planning must be tractable.

The causal graph and domain transition graphs for an RB task Π are defined exactly as for FDR. By the **black causal graph** CG^{B}_Π of Π, denote the sub-graph of CG_Π induced by the black variables. Say that an arc (d, d') is **relaxed side effects invertible**, **RSE-invertible** for short, if there exists an arc (d', d) with outside condition $\phi' \subseteq \phi \cup \psi$ where ϕ and ψ are the outside condition respectively outside effect of (d, d'). A variable v is RSE-invertible if all arcs in $DTG_\Pi(v)$ are RSE-invertible, and an RB task is RSE-invertible if all its black variables are.

Katz et al. (2013a) prove that plan generation for RSE-invertible RB tasks whose black causal graphs are acyclic is tractable. Specifically, plan existence in this setting is shown to be equivalent to relaxed plan existence, and Katz et al.'s algorithm is based on repairing a relaxed plan for Π.

Repairing Relaxed Plans

The algorithm UNRELAX(Π, π^+) for RSE-invertible RB tasks with acyclic black causal graphs, presented by Katz et al. (2013a) is depicted in Figure 2. It takes a relaxed plan π^+, which serves as a skeleton for incrementally constructing the red-black plan π. This is done by going over the actions of π^+ and inserting sequences of actions achieving black preconditions (where needed). The sequences are found by the ACHIEVE procedure, which constructs and solves (in polynomial time) an FDR planning task Π^{B}, whose initial state consists of the current black values, and whose goal is to achieve the black preconditions in question.

A major weakness of the UNRELAX algorithm is overestimation, incurred by following the decisions of a relaxed plan. Consider our example in Figure 1, and let $V^{\mathsf{B}} = \{R\}$. Say the relaxed plan π^+ starts with *move*$(1, 2)$, *move*$(2, 3)$, *take*$(2, A)$, *open*$(3, 4, A)$. Then a call to ACHIEVE(π, g) before *take*$(2, A)$ will insert *move*$(3, 2)$, and another call before *open*$(3, 4, A)$ will insert *move*$(2, 3)$. As Katz et al. (2013a) point out, similar phenomena occur massively in standard IPC benchmarks.

In Elevators, a relaxed plan tends to use actions of the form $move(c, X)$ where c is the current lift position and X are the floors that need to be visited. Given this input, whenever the lift needs to go from X to Y, a red-black plan uses $move(X, c), move(c, Y)$ instead of $move(X, Y)$. This becomes much worse still in Visitall. If, for example, in the current state we're located in the right bottom corner of a grid, then the relaxed plan is likely to visit the grid in a breadth-first fashion, going outwards in all directions from that corner. Given this, during red-black planning, as we reach for example the top right corner, instead of just moving one step to the left to the next grid cell, we move all the way back to the bottom before moving out from there again.

Going back to Elevators, if $board/leave$ actions are not up front, the red-black plan uselessly move elevators back and forth without taking care of any passengers. Also, since the relaxed plan is free to choose any $board/leave$ actions, it may decide to make all $board$s with the same capacity precondition (the initially true one for that lift). This forces the red-black plan to achieve the desired capacity by applying useless instances of $board/leave$.

To illustrate another extreme, related to resource-usage, consider a Logistics example with a star-shaped map over nodes m (middle) and $o_1, \ldots, o_N$ (outside nodes), with N trucks and N packages, all initially located at m, with the goal of getting each package p_i to its individual goal o_i. An optimal relaxed plan can use a single truck. Starting from this, the red-black plan uses a single truck as well, not making use of the much cheaper option to employ all N trucks.

Red-Black Planning

To tackle these issues, we escape the limitation of restricting ourselves to the actions from the relaxed plan. It turns out we can make do with a much weaker restriction, namely that to the red facts used by the relaxed plan.

Algorithm

Pseudo-code for our algorithm is shown in Figure 3.[1]

Theorem 1 *Let* $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G\rangle$ *be an RSE-invertible RB planning task with acyclic black causal graph, let* π^+ *be a relaxed plan for* Π*, and let* $R^+ = G[V^{\mathsf{R}}] \cup \bigcup_{a\in\pi^+} \mathsf{pre}(a)[V^{\mathsf{R}}]$*. Then the action sequence* π *returned by* REDBLACKPLANNING(Π, R^+) *is a plan for* Π*.*

The algorithm maintains two monotonically increasing sets of variable values: R is the set of all *currently achieved* red variable values; B is the set of all black variable values *currently reachable under* R. Both R and B are maintained by the UPDATE procedure. For $v \in V^{\mathsf{B}}$, $DTG_\Pi(v)|_{R\cup B}$ is obtained as follows. Let G be the subgraph of $DTG_\Pi(v)$ obtained by removing all arcs whose outside condition is not contained in $R \cup B$. The graph $DTG_\Pi(v)|_{R\cup B}$ is obtained from G by removing all vertices (and incident arcs) that are not reachable from $I[v]$. Abusing notation, we will

[1] Note that different relaxed plans could use different sets of red facts R^+, i. e., our approach is unrelated to landmarks.

Algorithm : REDBLACKPLANNING(Π, R^+)

main
// $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G\rangle$
global R, $B \leftarrow \emptyset$, $\pi \leftarrow \langle\rangle$
UPDATE()
while $R \not\supseteq R^+$
do
- $A' = \{a \in A \mid \mathsf{pre}(a) \subseteq B \cup R, \mathsf{eff}(a) \cap (R^+ \setminus R) \neq \emptyset\}$
- Select $a \in A'$
- **if** $\mathsf{pre}(a)[V^{\mathsf{B}}] \not\subseteq I[\![\pi]\!]$
  - **then** $\pi \leftarrow \pi \circ$ ACHIEVE($\mathsf{pre}(a)[V^{\mathsf{B}}]$)
- $\pi \leftarrow \pi \circ \langle a\rangle$
- UPDATE()

if $G[V^{\mathsf{B}}] \not\subseteq I[\![\pi]\!]$
then $\pi \leftarrow \pi \circ$ ACHIEVE($G[V^{\mathsf{B}}]$)
return π

procedure UPDATE()
$R \leftarrow I[\![\pi]\!][V^{\mathsf{R}}]$
$B \leftarrow B \cup I[\![\pi]\!][V^{\mathsf{B}}]$
for $v \in V^{\mathsf{B}}$, ordered topologically by the black causal graph
do $B \leftarrow B \cup DTG_{\Pi}(v)|_{R \cup B}$

procedure ACHIEVE(g)
$I^{\mathsf{B}} \leftarrow I[\![\pi]\!][V^{\mathsf{B}}]$
$G^{\mathsf{B}} \leftarrow g$
$A^{\mathsf{B}} \leftarrow \{a^{\mathsf{B}} \mid a \in A, a^{\mathsf{B}} = \langle \mathsf{pre}(a)[V^{\mathsf{B}}], \mathsf{eff}(a)[V^{\mathsf{B}}]\rangle,$
$\mathsf{pre}(a) \subseteq R \cup B, \mathsf{eff}(a)[V^{\mathsf{B}}] \subseteq B\}$
$\langle a'^{\mathsf{B}}_1, \ldots, a'^{\mathsf{B}}_k\rangle \leftarrow$ an FDR plan for $\Pi^{\mathsf{B}} = \langle V^{\mathsf{B}}, A^{\mathsf{B}}, I^{\mathsf{B}}, G^{\mathsf{B}}\rangle$
return $\langle a'_1, \ldots, a'_k\rangle$

Figure 3: Our red-black planning algorithm. $R^+ = G[V^{\mathsf{R}}] \cup \bigcup_{a \in \pi^+} \mathsf{pre}(a)[V^{\mathsf{R}}]$ where π^+ is a relaxed plan for Π.

use $DTG_{\Pi}(v)|_{R \cup B}$ to denote both the DTG sub-graph and the set of vertices (variable values) of that graph.

We start by showing that, for $R \not\supseteq R^+$, we always have $A' \neq \emptyset$. This is done with the help of the relaxed plan π^+. Let $a_i \in \pi^+$ be the action with the minimal index i such that $\mathsf{eff}(a_i) \cap (R^+ \setminus R) \neq \emptyset$. Thus, for $1 \leq j \leq i-1$, $\mathsf{eff}(a_j) \cap (R^+ \setminus R) = \emptyset$. Assume to the contrary that there exists $v \in \mathcal{V}(\mathsf{pre}(a_i)) \cap V^{\mathsf{R}}$ such that $\mathsf{pre}(a_i)[v] \notin R$. But then $\mathsf{pre}(a_i)[v] \neq I[v]$ and thus there exists $1 \leq j \leq i-1$ such that $\mathsf{eff}(a_j)[v] = \mathsf{pre}(a_i)[v] \in R^+$, giving us $\mathsf{eff}(a_j) \cap (R^+ \setminus R) \neq \emptyset$. Therefore we have $\mathsf{pre}(a_i)[V^{\mathsf{R}}] \subseteq R$. To see that $\mathsf{pre}(a_i)[V^{\mathsf{B}}] \subseteq B$, note that $a_1 \cdot \ldots \cdot a_{i-1}$ correspond to a path in $DTG_{\Pi}(v)$ for each black variable $v \in \mathcal{V}(\mathsf{pre}(a_i)) \cap V^{\mathsf{B}}$ that passes the value $\mathsf{pre}(a_i)[v]$ and uses only actions with outside conditions in $R \cup B$. Therefore $\mathsf{pre}(a_i) \subseteq R \cup B$ and we have the desired $a_i \in A'$.

We continue with the **while** loop. Consider an iteration of the loop. Any red preconditions of the selected action $a \in A'$ are true in the current state $I[\![\pi]\!]$ by the definition of A'. For the unsatisfied black preconditions $g = \mathsf{pre}(a)[V^{\mathsf{B}}]$ we have $g \subseteq B$ and thus they are tackled by ACHIEVE(g), solving an FDR task Π^{B} with goal g. Assume for the moment that this works correctly, i. e., the returned action sequence π^{B} is red-black applicable in the current state $I[\![\pi]\!]$ of our RB task Π. Π^{B} ignores the red variables, but effects on these cannot hurt anyway, so a is applicable in $I[\![\pi \circ \pi^{\mathsf{B}}]\!]$. Since $\mathsf{eff}(a) \cap (R^+ \setminus R) \neq \emptyset$, $|R^+ \setminus R|$ decreases by at least 1 at each iteration, so the **while** loop terminates. Upon termination, we have (i) $R^+ \subseteq I[\![\pi]\!][V^{\mathsf{R}}] = R$, and thus (ii) $G[V^{\mathsf{B}}] \subseteq B$. Then, calling ACHIEVE($G[V^{\mathsf{B}}]$) (if needed) will turn π into a plan for our RB task Π.

We now conclude the proof of Theorem 1 by showing that: (i) Π^{B} is well-defined; (ii) Π^{B} is solvable and a plan can be generated in polynomial time; and (iii) any plan π^{B} for Π^{B} is, in our RB task Π, applicable in $I[\![\pi]\!]$.

For (i), we show that all variable values occuring in Π^{B} are indeed members of the respective variable domains. This is obvious for I^{B} and A^{B}. It holds for $G^{\mathsf{B}} = \mathsf{pre}(a)[V^{\mathsf{B}}]$ as $a \in A'$. For $G^{\mathsf{B}} = G[V^{\mathsf{B}}]$, once we have $R \supseteq R^+$, π^+ corresponds to a path in $DTG_{\Pi}(v)$ for each black variable $v \in V^{\mathsf{B}} \cap \mathcal{V}(G)$ through $G[v]$, and thus $G[V^{\mathsf{B}}] \subseteq B$.

For (iii), we have $\mathsf{pre}(a) \subseteq R \cup B$ for all actions where $a^{\mathsf{B}} \in A^{\mathsf{B}}$. This implies that the red preconditions of all these a are true in the current state $I[\![\pi]\!]$. So applicability of π^{B} in $I[\![\pi]\!]$, in Π, depends on the black variables only, all of which are contained in Π^{B}.

To show (ii), we define a new planning task by restricting the set of actions. We show that this new planning task is solvable, and a plan can be generated in polynomial time; the same then obviously follows for Π^{B}. Let $\Pi^{\mathsf{B}}_{\pi} = \langle V^{\mathsf{B}}, A^{\mathsf{B}}_{\pi}, I^{\mathsf{B}}, G^{\mathsf{B}}\rangle$ be the planning task obtained from Π^{B} by (1) setting the actions to $A^{\mathsf{B}}_{\pi} = \{a^{\mathsf{B}} \mid a \in A, a^{\mathsf{B}} = \langle \mathsf{pre}(a)[V^{\mathsf{B}}], \mathsf{eff}(a)[V^{\mathsf{B}}]\rangle, \mathsf{pre}(a) \cup \mathsf{eff}(a) \subseteq I[\![\pi^+]\!]\}$; and (2) by restricting the variable domains to the values in $I[\![\pi^+]\!]$. It is easy to see that Π^{B}_{π} is well-defined. As $I[\![\pi^+]\!] \subseteq R \cup B$, we obviously have $A^{\mathsf{B}}_{\pi} \subseteq A^{\mathsf{B}}$ as advertized.

We next show that the domain transition graphs are strongly connected. Observe that all values in $DTG_{\Pi^{\mathsf{B}}_{\pi}}(v)$ are, by construction, reached from $I[v]$ by a sequence of arcs (d, d') induced by actions in π. So it suffices to prove that every such arc has a corresponding arc (d', d) in $DTG_{\Pi^{\mathsf{B}}_{\pi}}(v)$. Say $v \in V^{\mathsf{B}}$, and (d, d') is an arc in $DTG_{\Pi^{\mathsf{B}}_{\pi}}(v)$ induced by a^{B} where $a \in \pi$. Since (d, d') is RSE-invertible in Π, there exists an action $a' \in A$ inducing an arc (d', d) in $DTG_{\Pi}(v)$ whose outside condition is contained in $\mathsf{pre}(a) \cup \mathsf{eff}(a)$. Since, obviously, $\mathsf{pre}(a) \cup \mathsf{eff}(a) \subseteq I[\![\pi^+]\!]$, we get $\mathsf{pre}(a') \subseteq I[\![\pi^+]\!]$. Now, a' can have only one black effect (otherwise, there would be a cycle in the black causal graph) so $\mathsf{eff}(a')[V^{\mathsf{B}}] = \{(v, d)\}$ which is contained in $I[\![\pi^+]\!]$. Thus $a'^{\mathsf{B}} \in A^{\mathsf{B}}_{\pi}$, and (d', d) is an arc in $DTG_{\Pi^{\mathsf{B}}_{\pi}}(v)$ as desired.

This suffices because plan generation for FDR with acyclic causal graphs and strongly connected DTGs is tractable: Precisely, every such task is solvable, and a plan *in a succinct plan representation* can be generated in polynomial time. This is a direct consequence of Theorem 23 of Chen and Gimenez (2008) and Observation 7 of Helmert (2006). (The succinct representation consists of recursive macro actions for pairs of values with each variable's DTG; it is required as plans may be exponentially long.) This concludes the proof of Theorem 1.

Over-Estimation

Going back to the issues outlined in the previous section, let π^+ be a relaxed plan and R^+ be the set of facts obtained from π^+ as in Theorem 1. In Elevators, when setting the lift locations to be black variables and the rest to be red, R^+ will consist of passenger initial, intermediate (in some lifts) and

goal locations, as well as of lift capacities required along π^+. The actions achieving these facts are $board/leave$; these are the actions selected by the main loop. All $move$ actions are added by the $ACHIEVE(g)$ procedure, so the lifts will not move back and forth without a purpose. This covers the first two issues of the Elevators domain. For the third issue, assume that all $board$ actions in π^+ are preconditioned by the initially true capacity of that lift (c_l). Then all $leave$ actions in π^+ will be preconditioned by the $c_l - 1$ capacity, and c_l and $c_l - 1$ are the only capacity-related facts in R^+. As c_l is true in the initial state, and $c_l - 1$ is achieved by the first $board$ action into the respective lift, after that action there are no more capacity-related facts in $R^+ \setminus R$. Thus action selection in the main loop will be based exclusively on following red facts related to the passenger locations, solving the third issue.

In Visitall, painting the robot black and the locations red will cause the main loop to select some achieving action for every unvisited locations. In our implementation, these actions are selected in a way minimizing the cost of achieving their preconditions, solving this issue as well.

The star-shaped Logistics example remains a problematic case. Painting the trucks black and the packages red will cause the main loop to follow the positions of the packages. Assume now that the relaxed plan π^+ uses only the truck t. R^+ will then include the initial and goal positions of each package, as well as the fact $in(p, t)$ for each package p. Thus, as before, the red-black plan will use only the same single truck t. It remains an open question how to resolve this; perhaps low-conflict relaxed plans (Baier and Botea 2009) could deliver input better suited to this purpose.

Implementation

We adopt from Katz et al. (2013a) a simple optimization that tests, in every call to the heuristic, whether the red-black plan generated is actually a real plan, and if so, stop the search. We denote this technique with **S** for "stop" and omit the "**S**" when not using it. Also, we need a technique to choose the red variables. Similarly to Katz et al. (2013a), we start by painting red all variables that are not RSE-invertible. Further, we paint red all causal graph leaves because that does not affect the heuristic. From the remaining variable set, we then iteratively pick a variable v to be painted red next, until there are no more arcs between black variables. Our techniques differ in how they choose v:

- **A**: Select v with the maximal number $A(v)$ of incident arcs to black variables; break ties by smaller domain size.
- **C**: Select v with the minimal number $C(v)$ of conflicts, i. e., relaxed plan actions with a precondition on v that will be violated when executing the relaxed plan with black v.
- **C[N]**: Extends **C** by sampling N random states, then select v with the minimal average number of conflicts.
- **CA[p]**: Interpolation between **C** (with $p = 0$) and **A** (with $p = 1$). Select v with the maximal value $P(v) := p*\hat{A}(v)+(1-p)*(1-\hat{C}(v))$, where $\hat{A}(v)$ and $\hat{C}(v)$ represent the number of incident edges and number of conflicts, respectively.
- **L**: Select v with highest level in the causal graph heuristic (Helmert 2004).

The first two techniques were introduced by Katz et al. (2013a). The intuition behind **A** is to minimize the number (and the domain sizes) of red variables. The intuition behind **C** is for the least important variables to be red. However, **C** depends on a particular relaxed plan, and in cases when the relaxed plan chose not to exploit the available resources, these will have no conflicts. In an attempt to make the red variable selection more stable, we propose the technique **C[N]**, which samples N random states and finds a relaxed plan for each state in the sample. The choice of the next variable v is then made based on the average number of conflicts of v. The idea behind **CA[p]** is to interpolate between **C** and **A**, aiming at getting the best of both worlds. A variable v maximizing $P(v) := p*\hat{A}(v)+(1-p)*(1-\hat{C}(v))$ is chosen to be red. Note that $\hat{A}(v)$ is the number of incident edges of v divided by the maximal number of incident edges among all invertible variables, and $\hat{C}(v)$ is the number of conflicts of v divided by the maximal number of conflicts of all variables. Naturally, for $p = 1$ we obtain the method **A**, while for $p = 0$ we get **C**. However, **A** and **C** of Katz et al. have different tie breaking. Our implementation of **CA[p]** adopts the tie breaking of **A**. The last technique is causal graph based. **L** aims at painting the "servant" variables, those that change their value in order to support other variables, black.

Experiments

The experiments were run on Intel(R) Xeon(R) CPU X5690 machines, with time (memory) limits of 30 minutes (2 GB). We ran all STRIPS benchmarks from the IPC. Since the 2008 domains were run also in 2011, we omit those to avoid a bias on these domains. For the sake of simplicity we consider uniform costs throughout (i. e., we ignore action costs where specified). Furthermore, since our techniques do not do anything if there are no RSE-invertible variables, we omit instances in which that is the case (and domains where it is the case for all instances, specifically Airport, Freecell, Parking, Pathways-noneg, and Openstacks domains).

Our main objective in this work is to improve on the relaxed plan heuristic, so we compare performance against that heuristic. Precisely, we compare against this heuristic's implementation in Fast Downward. We run a configuration of Fast Downward commonly used with inadmissible heuristics, namely greedy best-first search with lazy evaluation and a second open list for states resulting from preferred operators (Helmert 2006). To enhance comparability, we did not modify the preferred operators, i. e., all our red-black relaxed plan heuristics simply take these from the relaxed plan. We also run an experiment with a single open list, not using preferred operators.

We compare to the best performing configuration (**F**) of the red-black plan heuristic proposed by Katz et al. (2013a), that we aim to improve in the current work. We denote our techniqe of red facts following by **R**. A comparison to the approach of Keyder et al. (2012) was performed as well. Precisely, we use two variants of this heuristic, that we refer

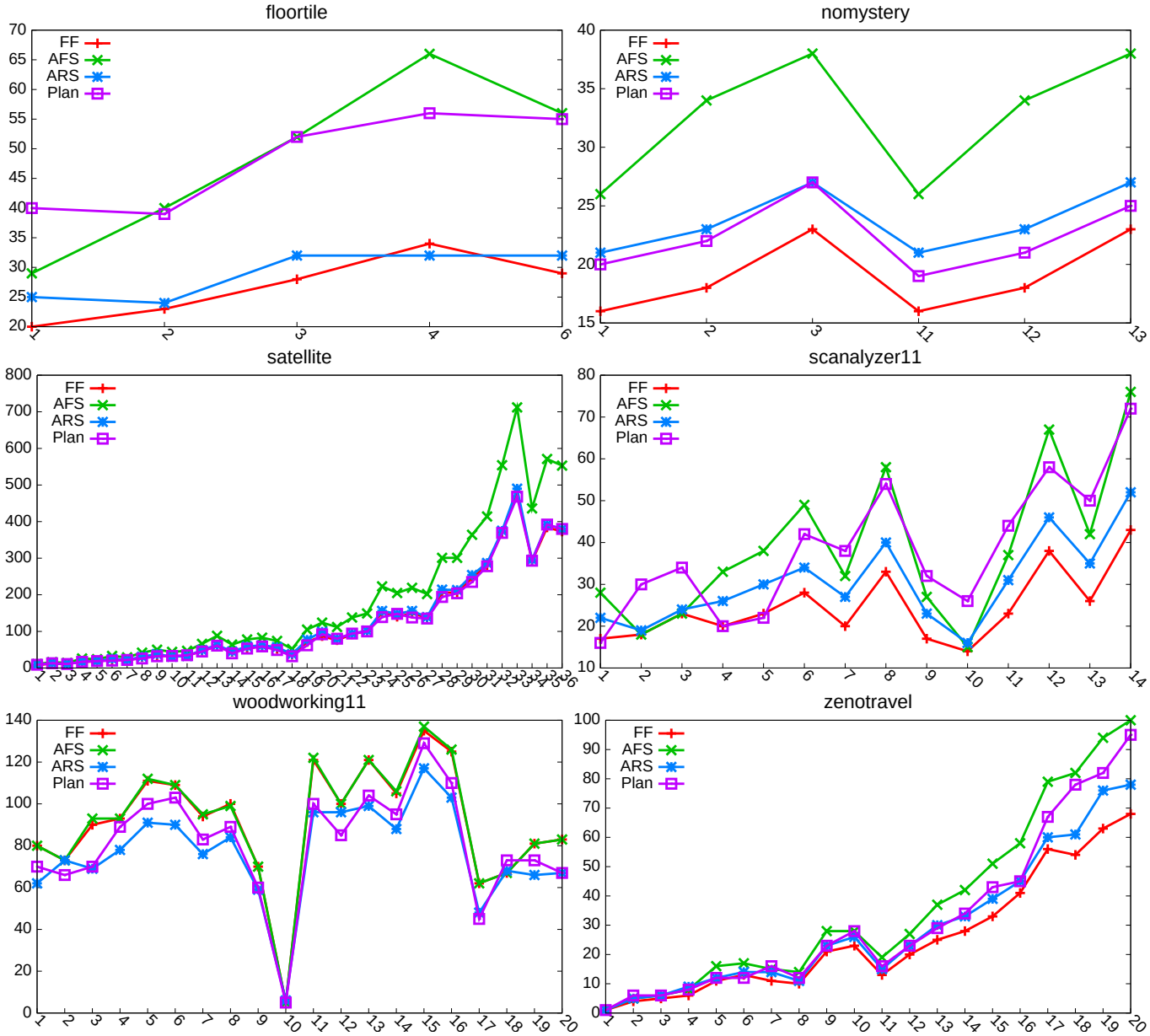


Figure 4: Initial state heuristic values and best plan found.

to as **Keyd**'12 and **Keyd**'13. **Keyd**'12 is the overall best-performing configuration from the experiments as published at ICAPS'12. **Keyd**'13 is the overall best-performing configuration from a more recent experiment run by Keyder et al. (unpublished; private communication).

We ran 20 variants of our own heuristic, switching **S** on/off, and running one of **A**, **C**, **C[N]** for N $\in \{5, 25, 100\}$, **CA[p]** for p $\in \{0, 0.25, 0.5, 0.75\}$, or **L**. We measured: coverage (number of instances solved); total runtime; search space size as measured by the number of calls to the heuristic function; initial state accuracy as measured by the difference between the heuristic value for I and the length of the best plan found by any planner in the experiment; as well as heuristic runtime as measured by search time divided by the number of calls to the heuristic function. Table 1 provides an overview pointing out the main observations. As our objective here is to find better red-black plans, Figure 4 shows the initial state heuristic values for selected domains.

Consider first the coverage data in Table 1. Not using preferred operators illuminates the advantages of our new techniques quite drastically: Whereas the previous heuristic **F** *decreases* coverage over the FF heuristic, our new technique **R** *increases* it. The margins are substantial in particular for the best performing configurations **AR** vs. **AF**.

Consider now the coverage data when using preferred operators. Focussing on significant differences in specific domains reveals that the best performing configuration **ARS** improves over the best performing configuration **AFS** of Katz et al. (2013a), solving 2 more tasks in Floortile and 8 more tasks in Nomystery. Comparing **CRS** to **CFS** shows improvement of 12 tasks in Elevators, 2 in Floortile, 8 in Nomystery and 4 tasks in Transport. On the other hand,

		Coverage																				Time FF/OWN (med)				H-time OWN/FF							
		No Pref. Ops.					Keyd			AS		CS		C[N]RS		CA[p]RS			LS			AS		CS		AF		AR		CF		CR	
	#	FF	AF	AR	CF	CR	'12	'13	FF	F	R	F	R	5	100	0	0.25	0.75	F	R		F	R	F	R	med	max	med	max	med	max	med	max
barman	20/20	15	16	16	17	2	4	18	19	20	19	20	5	6	6	13	9	19	20	19		7,96	22,0	138	1,21	1,24	1,56	1,00	1,48	1,01	1,21	1,11	1,67
blocks	35/35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35		1,00	1,00	1,00	1,00	0,93	2,74	0,84	3,44	0,99	4,82	0,93	4,84
depot	22/22	15	14	15	14	15	21	21	18	19	19	19	19	19	19	19	19	19	19	19		1,00	1,00	1,00	1,00	1,06	3,73	1,27	4,15	1,11	3,73	1,15	4,15
driverlog	20/20	18	16	18	17	18	20	20	20	20	20	20	20	20	20	20	20	20	20	20		1,00	1,00	1,00	1,00	1,08	2,96	1,05	1,46	1,00	9,85	1,01	1,66
elevators	20/20	17	14	13	2	11	19	18	20	20	20	8	20	20	20	20	20	20	14	19		0,92	0,50	0,08	1,34	1,09	1,39	1,82	2,23	1,27	1,73	1,47	1,67
floortile	20/20	4	6	3	6	3	20	20	5	5	7	5	7	7	7	7	7	7	5	7		0,60	0,20	0,54	0,20	1,83	7,10	2,34	2,46	2,00	7,10	2,27	2,34
grid	5/5	4	3	4	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5		0,93	0,94	0,83	0,85	1,21	1,71	1,41	1,82	1,24	1,71	1,37	2,08
gripper	20/20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20		1,00	1,00	1,00	1,00	4,17	6,90	3,91	6,95	4,17	6,90	3,91	6,95
logistics00	28/28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28		1,00	1,00	1,00	1,00	0,96	1,30	1,15	1,61	0,96	1,30	1,15	1,61
logistics98	35/35	22	5	35	5	35	35	35	34	35	35	35	35	35	35	35	35	35	35	35		1,29	1,32	1,20	1,23	1,19	1,79	1,40	4,05	1,19	2,11	1,48	4,05
miconic	150/150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150		1,00	1,00	1,00	1,00	1,05	1,44	2,00	4,47	1,04	1,44	2,00	4,47
mprime	35/35	30	31	30	29	30	35	35	35	35	35	34	35	35	35	35	35	35	35	35		0,79	0,80	0,80	0,79	1,00	3,11	1,00	10,2	1,02	4,36	1,04	9,03
mystery	28/30	17	17	17	17	17	19	19	16	17	16	16	16	16	16	16	16	16	16	16		0,97	0,93	0,95	0,93	1,00	2,50	1,00	12,0	1,12	2,77	1,01	3,13
nomystery	20/20	8	7	14	7	14	6	6	10	6	14	6	14	14	14	14	14	14	6	14		0,88	1,20	0,88	1,15	1,03	1,62	1,87	26,2	1,23	1,68	1,87	26,2
parcprinter	13/20	4	6	4	6	4	3	1	13	13	13	13	13	13	13	13	13	13	13	13		1,00	1,00	1,00	1,00	1,01	12,8	1,03	14,4	1,01	12,8	1,03	14,4
pegsol	20/20	20	20	20	20	20	19	20	20	20	20	20	20	20	20	20	20	20	20	20		1,00	1,00	1,00	1,00	1,00	3,22	1,57	34,3	1,00	3,22	1,55	16,1
Pipes-notank	40/50	20	18	18	18	18	33	33	33	33	32	33	32	32	32	32	32	32	33	32		0,85	1,00	0,93	1,00	1,08	3,36	1,09	14,67	1,20	15,6	1,22	24,1
Pipes-tank	40/50	14	16	12	16	13	29	29	29	29	30	29	30	29	30	29	30	30	29	30		0,92	0,97	0,70	0,95	1,33	2,71	1,40	2,53	1,07	2,04	1,34	3,01
Psr-small	50/50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50		1,00	1,00	1,00	1,00	1,00	5,20	1,00	6,10	1,00	4,20	1,00	4,90
rovers	40/40	23	16	25	17	25	40	40	40	40	40	40	40	40	40	40	40	40	40	40		1,00	0,92	1,00	0,95	1,15	2,08	1,64	4,79	1,29	2,66	1,70	5,00
satellite	36/36	23	22	28	22	28	34	34	36	36	36	36	35	35	35	35	35	35	36	36		1,00	1,23	1,00	1,20	1,25	2,01	1,57	4,39	1,18	1,77	1,56	3,32
scanalyzer	14/20	10	12	14	10	10	14	14	14	14	14	14	14	14	14	14	14	14	14	14		0,84	1,13	0,98	1,01	1,12	2,39	2,03	4,21	1,11	2,48	1,00	2,05
sokoban	20/20	19	19	19	18	19	17	16	19	19	19	18	19	19	19	19	19	19	19	19		0,55	0,41	0,26	0,38	1,92	5,28	1,98	3,17	4,15	5,96	2,33	3,74
tidybot	20/20	15	14	13	16	13	15	12	15	13	12	17	13	14	15	14	15	12	14	10		0,83	0,59	1,08	0,68	1,66	2,44	1,79	3,30	1,61	2,60	1,67	2,83
tpp	30/30	20	15	20	15	20	30	30	30	29	30	29	30	30	30	30	30	30	29	30		1,00	0,98	1,00	0,95	1,04	1,93	1,19	2,32	1,07	1,32	1,13	1,76
transport	20/20	0	0	0	1	0	11	11	10	10	10	8	12	11	12	13	14	10	7	11		0,99	0,54	1,38	0,79	1,12	1,91	1,85	2,33	0,36	1,30	1,24	1,59
trucks	30/30	16	15	16	16	14	14	14	18	20	20	19	18	18	18	18	18	18	17	18		1,00	1,00	0,53	0,76	1,13	6,43	1,29	10,43	1,18	3,67	1,25	3,11
visitall	20/20	5	3	17	3	17	20	20	3	20	20	20	20	20	20	20	20	20	20	20		17,5	17,5	17,5	17,5	2,79	3,12	17,9	23,4	2,57	3,19	19,1	23,7
woodworking	20/20	2	2	3	2	3	20	20	20	20	20	20	20	20	20	20	20	20	20	20		0,65	0,82	0,66	0,71	0,92	1,00	0,82	1,00	0,76	1,00	1,07	1,31
zenotravel	20/20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20		1,00	1,00	1,00	1,00	1,14	1,63	1,36	2,08	1,14	1,63	1,34	1,76
$\sum$	891/926	644	610	677	601	656	786	794	785	801	**809**	787	795	795	798	804	803	806	789	805													

	Evaluations FF/OWN																		Init	|OWN-P| / |FF-P|			
	Keyd'12			Keyd'13			AFS			ARS			CFS			CRS			AS (med)		CS (med)	
	min	med	max	min	med	max	min	med	max	min	med	max	min	med	max	min	med	max	F	R	F	R
barman	0.01	1.11	16.1	0.09	1.84	1349	0.80	8.40	433	0.88	34.0	484	1.44	135	514	0.00	1.16	347	0.99	0.99	0.96	1.02
blocks	0.79	2.84	38.0	0.83	3.00	39.9	0.46	0.92	2.77	0.11	0.97	7.00	0.40	0.99	4.83	0.31	1.10	7.00	1.00	1.00	0.52	0.73
depot	0.59	5.95	4551	0.00	7.38	8916	0.15	1.03	46.5	0.12	1.75	7.31	0.15	1.03	46.5	0.12	1.75	7.31	0.86	0.92	0.86	0.92
driverlog	0.11	1.61	91.1	0.02	1.26	60.2	0.04	0.96	15.5	0.01	1.00	17.0	0.04	1.00	10.5	0.03	1.00	17.0	0.54	0.77	0.54	0.64
elevators	0.22	1.34	37.2	0.22	1.05	37.2	0.54	1.00	5.84	0.49	1.03	7.49	0.02	0.11	0.67	0.37	2.09	14.8	0.94	1.00	1.40	0.92
floortile	986	13852	24221	761	14117	18486	0.54	0.89	237	0.30	0.41	3.91	0.54	0.89	237	0.30	0.41	3.91	0.06	0.88	0.06	0.88
grid	0.32	1.80	4.29	0.65	1.59	2.19	0.93	1.00	1.07	0.98	1.00	1.65	0.16	1.00	1.15	0.98	1.00	1.00	1.00	1.00	0.92	1.00
gripper	0.50	0.65	1.00	0.45	0.94	1.00	1.38	4.29	7.02	1.50	4.00	7.07	1.38	4.29	7.02	1.50	4.00	7.07	1.17	1.00	1.17	1.00
logistics00	1.05	1.56	2.22	0.97	1.52	2.06	10.0	137	516	10.0	137	516	10.0	137	516	10.0	137	516	1.83	0.00	1.83	0.00
logistics98	0.08	2.69	59.5	0.05	2.48	58.6	24.0	448	36919	24.0	448	36919	24.0	448	36919	24.0	448	36919	2.71	0.00	2.71	0.00
miconic	0.60	1.68	2.69	0.60	1.33	2.36	3.00	253	925	3.00	253	925	3.00	253	925	3.00	253	925	5.67	0.00	5.67	0.00
mprime	0.03	1.86	24.8	0.50	1.83	24.8	1.00	1.00	1.02	0.88	1.00	2.47	0.02	1.00	4.80	0.37	1.00	2.47	1.00	1.00	0.67	1.00
mystery	0.00	1.23	1.6E+7	0.00	1.08	1.6E+7	0.83	1.00	1.00	1.00	1.00	3.45	0.15	1.00	3.00	0.38	1.00	3.45	1.00	1.00	1.00	1.00
nomystery	0.44	1.23	4.09	1.25	1.98	45.8	0.01	0.66	2.19	0.45	2.99	75.0	0.01	0.66	2.19	0.45	2.99	75.0	2.88	0.25	2.88	0.25
parcprinter	0.00	0.01	0.30	0.51	0.51	0.51	0.01	1.04	1708	0.01	1.07	1913	0.01	1.04	1708	0.01	1.07	1913	1.00	1.00	1.00	1.00
pegsol	0.03	1.18	26.2	0.05	1.30	103	0.14	1.02	7.35	0.09	1.29	481	0.10	1.00	21.8	0.09	1.37	21.4	1.00	1.07	1.00	1.07
Pipes-notank	0.02	1.01	80.2	0.04	0.69	73.9	0.78	1.00	4.74	0.20	3.08	73.8	0.69	1.02	17.0	0.35	3.61	73.8	1.00	1.10	1.00	1.08
Pipes-tank	0.01	0.85	453	0.02	0.84	1601	0.14	1.00	70.8	0.17	1.11	4.13	0.12	1.00	70.8	0.19	1.15	5.00	1.00	1.06	1.00	1.05
Psr-small	0.19	1.00	11.2	0.18	0.96	6.38	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
rovers	0.55	1.04	2.50	0.56	1.02	2.33	0.44	1.15	6.33	0.26	1.18	6.00	0.58	1.20	6.33	0.69	1.22	6.00	0.45	0.50	0.46	0.50
satellite	0.54	1.23	90.3	0.63	1.11	90.3	0.61	1.12	3404	3.21	55.5	32862	0.61	1.18	3404	3.21	51.0	32862	8.00	1.40	8.00	1.50
scanalyzer	0.89	3.53	137	0.31	2.91	11.80	0.33	1.04	2.08	1.20	2.46	171	0.06	1.15	10.1	0.78	1.24	51.6	0.45	0.67	0.36	1.00
sokoban	0.01	1.05	3.31	0.02	1.47	65.6	0.10	1.00	109	0.11	0.91	5.67	0.01	1.05	109	0.01	0.86	5.67	1.00	0.94	0.39	0.90
tidybot	0.21	0.83	2.96	0.00	1.01	2.92	0.96	1.05	1.73	0.48	1.08	7.76	0.19	1.05	2.49	0.19	0.99	2.07	1.00	1.00	1.00	0.99
tpp	0.41	0.92	1.62	0.35	0.81	1.62	0.05	0.76	22.0	0.20	1.13	22.0	0.05	0.76	22.00	0.20	1.13	22.0	0.57	0.83	0.57	0.83
transport	0.34	1.59	5.79	0.59	1.69	3.58	0.36	1.18	2.74	0.98	1.41	3.41	0.21	0.64	2.80	0.58	1.03	12.35	0.99	1.00	1.34	0.57
trucks	0.03	1.17	760	0.04	0.95	4.61	0.02	1.01	38.6	0.14	1.22	28.1	0.02	0.61	92.5	0.01	0.86	32.0	0.53	1.00	0.25	0.73
visitall	4.19	6.18	67.3	4.98	5.70	71.4	15508	18790	409512	15508	18790	409512	15508	18790	409512	15508	18790	409512	26.4	0.00	26.4	0.00
woodworking	0.83	227	452	0.83	225	452	0.92	1.00	5.00	0.90	1.13	2.50	0.92	1.00	5.00	0.90	1.13	2.50	1.00	0.70	1.00	0.40
zenotravel	0.50	1.19	4.28	0.50	0.90	5.86	0.53	1.08	6.08	0.92	1.38	2.88	0.53	1.08	6.08	0.92	1.38	2.88	1.00	0.40	1.00	0.40

Table 1: Selected results on IPC STRIPS benchmarks with at least one RSE-invertible variable, omitting IPC 2008 to avoid domain duplication, and ignoring action costs. # is the number of such instances per domain/overall per domain. Keyd'12 and Keyd'13 are heuristics by Keyder et al. (2012), FF is Fast Downward's relaxed plan heuristic, and A, C, C[N], CA[p], L, F, R, and S refer to our implementation options. Time is total runtime, Evaluations is the number of calls to the heuristic, and H-time is the average runtime taken by such a call. Init is the heuristic value for the initial state and P is the length of the best plan found. Data for these measures are shown for F and R configs in terms of per-task ratios against FF as indicated. For the purpose of measuring H-time the S option was switched off. We show features (median, min, max) of that ratio's per-domain distribution on the instances solved by both planners involved.

CRS loses 15 in Barman and 4 in Tidybot. Switching to **C[100]RS** seems to help a bit, gaining back 1 task in Barman and 2 in Tidybot. **CA[p]RS** does not seem to pay off overall, but it does have an interesting effect in Transport, solving more than the best of **ARS** and **CRS**. Another interesting observation is that **CA[0]RS** seems to perform better than **CRS**, solving 8 more tasks in Barman, 9 more tasks overall. For **L** the picture is similar to **C**, except for the Barman domain, where **LRS** loses only 1 task. **Keyd** seem to perform well, with the most significant result in Floortile domain, where it solves all 20 tasks, an impressive result that is negatively balanced by the bad performance in Parcprinter.

The runtime data reveals several interesting observations. First, the runtime for **R** typically improves over the previous method **F**. The advantage is still more often on FF's side, yet not as often as for **F**. Focusing on **A**, observe that, despite the much better runtime in Barman, the number of solved tasks for **ARS** is smaller than that for **AFS**. The opposite happens in Floortile, where despite the worse runtime, 2 more tasks are solved by **ARS**. For **C**, it's similar in Floortile and Transport, the latter having 4 more tasks solved, while the median runtime decreases considerably.

The heuristic runtime data in Table 1 shows that the slowdown of our approach is typically small.

Moving to median performance for search space size (Evaluations in Table 1) and comparing **R** to **F**, we note for **AS** a decrease in only 3 domains (with the largest one in Floortile), same performance in 8, and an increase in 19 domains, with the most considerable increase in Satellite, Nomystery, Barman, and Pipes-notank. For **CS** the picture is similar, with a decrease in only 5 domains (the largest in Barman and Floortile), same performance in 9, and an increase in 16 domains, with the most considerable increase in Satellite and Elevators. It turned out that **RS** allows for solving several additional instances of Satellite without search. As a result, the median search space reduction in 6 domains for **ARS** and 4 domains for **AFS**, and in 5 domains for **CRS** and **CFS**, is by 1–4 orders of magnitude. Taking a look beyond the median, the results for maximum show a reduction by 1–5 orders of magnitude in 14 domains for **ARS** and **AFS**, in 16 domains for **CRS** and 17 domains for **CFS**. For **Keyd**, the median search space reduction of more than one order of magnitude is only in two domains, namely Floortile and Woodworking. However, the results for maximum show a reduction of 1–7 orders of magnitude in 18 domains.

Finally, the last four columns in the lower part of Table 1 show the median of the absolute difference between the initial value and the length of the shortest plan found (P), relative to the absolute difference for the FF heuristic. The value is 0 when FF is inaccurate and the measured heuristic is exactly the length P. The value 1 means that both the measured heuristic and FF are equally (in)accurate, while values lower than 1 stand for the measured heuristic being more accurate than FF. We can see that for **AFS** the median estimate is worse than FF's in 7 domains, the same in 13, and better in 10 domains. For **ARS** it is worse in 4 domains, the same in 11, and better in 15 domains. For **CFS** the median estimate is worse in 9 domains, the same in 9, and better in 12 domains, and for **CRS** it is worse in 5 domains, the same in 7, and better in 18 domains. Comparing **R** to **F**, we see that, at least for median, the over-estimation is typically reduced, especially in Elevators, Logistics domains, Miconic, Nomystery, Satellite, Trucks, Visitall, Woodworking, and Zenotravel. Interestingly, the estimate in Floortile, despite being better than FF's, is much worse than **F**'s. Figure 4 takes a closer look into some of these domains, showing the heuristic values of **F**, **AFS**, and **ARS**, as well as the length of the best plan found. It shows the reduction in over-estimation and explains the increased performance in these domains. To some extent, it might explain what happens in Floortile as well: The *shape* of the **ARS** curve is more similar to the one that describes the length of the best plan found. So **ARS**, although less accurate than **AFS** (at least for median), may serve as better heuristic guidance.

Discussion

We devised a new way to compute red-black plan heuristics, improving over the previous method by relying less on relaxed plans. Our experiments confirm impressively (cf. the data when not using preferred operators) that this yields a far better heuristic than the previous red-black planning method, and that it substantially improves over a standard delete-relaxation heuristic. In the competitive setting with preferred operators, our new heuristic shows improvements in both the search space size, measured by the number of evaluations, and in the number of tasks solved.

It is not clear which message to take from our experiments with different methods for selecting red variables. As of now, the performance differences, though dramatic in individual domains, are not systematic. There is no good match to what we would have expected, based on our intuitions of the conceptual differences between these variable selection methods. It appears that, in the culprit domains, selecting one or another subset of variables makes a lot of difference for idiosyncratic reasons, and that one or another variable selection method just happens to select the right respectively wrong subset. This remains to be explored in more detail; it seems doubtful whether alternate methods will perform substantially better than the ones we already have.

Floortile points to what probably is a fundamental weakness of the red-black planning framework as it stands, especially in comparison to Keyder et al. (2012). The main issue in Floortile are dead-ends that go unrecognized by the relaxed plan heuristic (painting yourself "into a corner"). The red-black plan heuristics we have as off now are unable to help with this, at least to help to a dramatic extent, simply because relaxed plan existence implies red-black plan existence. That is very much not so for Keyder et al. At the heart of this is the RSE-invertibility assumption we currently make. It may be worthwhile to look into ways of getting rid of that restriction, while preserving tractability.

In conclusion, the red-black relaxed plan heuristics we devised do yield a significant improvement over the previous ones. Much remains to be done to fully exploit the potential of the red-black planning framework.

Acknowledgments. We thank Carmel Domshlak for discussions relating to this work.

References

Baier, J. A., and Botea, A. 2009. Improving planning performance using low-conflict relaxed plans. In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009)*. AAAI Press.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1–2):5–33.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69(1–2):165–204.

Cai, D.; Hoffmann, J.; and Helmert, M. 2009. Enhancing the context-enhanced additive heuristic with precedence constraints. In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 50–57. AAAI Press.

Chen, H., and Giménez, O. 2008. Causal graphs and structurally restricted planning. In Rintanen, J.; Nebel, B.; Beck, J. C.; and Hansen, E., eds., *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS 2008)*, 36–43. AAAI Press.

Fox, M., and Long, D. 2001. Stan4: A hybrid planning strategy based on subproblem abstraction. *The AI Magazine* 22(3):81–84.

Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning through stochastic local search and temporal action graphs. *Journal of Artificial Intelligence Research* 20:239–290.

Haslum, P. 2012. Incremental lower bounds for additive cost planning problems. In Bonet, B.; McCluskey, L.; Silva, J. R.; and Williams, B., eds., *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 74–82. AAAI Press.

Helmert, M., and Geffner, H. 2008. Unifying the causal graph and additive heuristics. In Rintanen, J.; Nebel, B.; Beck, J. C.; and Hansen, E., eds., *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS 2008)*, 140–147. AAAI Press.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In Koenig, S.; Zilberstein, S.; and Koehler, J., eds., *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, 161–170. Whistler, Canada: Morgan Kaufmann.

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

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Katz, M.; Hoffmann, J.; and Domshlak, C. 2013a. Red-black relaxed plan heuristics. In desJardins, M., and Littman, M., eds., *Proceedings of the 27th National Conference of the American Association for Artificial Intelligence (AAAI'13)*. Bellevue, WA, USA: AAAI Press. Forthcoming.

Katz, M.; Hoffmann, J.; and Domshlak, C. 2013b. Who said we need to relax *All* variables? In Borrajo, D.; Fratini, S.; Kambhampati, S.; and Oddi, A., eds., *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS 2013)*. AAAI Press. Forthcoming.

Keyder, E., and Geffner, H. 2008. Heuristics for planning with action costs revisited. In Ghallab, M., ed., *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08)*, 588–592. Patras, Greece: Wiley.

Keyder, E.; Hoffmann, J.; and Haslum, P. 2012. Semi-relaxed plan heuristics. In Bonet, B.; McCluskey, L.; Silva, J. R.; and Williams, B., eds., *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012)*. AAAI Press.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39:127–177.