Red-Black Relaxed Plan Heuristics

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

Carmel Domshlak
Technion
Haifa, Israel
dcarmel@ie.technion.ac.il

Abstract

Despite its success, the delete relaxation has significant pitfalls. Recent work has devised the *red-black planning* framework, where red variables take the relaxed semantics (accumulating their values), while black variables take the regular semantics. Provided the red variables are chosen so that red-black plan generation is tractable, one can generate such a plan for every search state, and take its length as the heuristic distance estimate. Previous results were not suitable for this purpose because they identified tractable fragments for red-black plan *existence*, as opposed to red-black plan *generation*. We identify a new fragment of red-black planning, that fixes this issue. We devise machinery to efficiently generate red-black plans, and to automatically select the red variables. Experiments show that the resulting heuristics can significantly improve over standard delete relaxation heuristics.

Introduction

In the *delete relaxation*, that we will also refer to as the *monotonic relaxation* here, state variables accumulate their values, rather than switching between them. This relaxation has turned out useful for a variety of purposes, in particular for satisficing planning (not giving an optimality guarantee), which we focus on here. The generation of (non-optimal) plans in monotonic planning is polynomial-time (Bylander 1994), enabling its use for the generation of (non-admissible) heuristic functions. Such *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 pitfalls in many important classes of planning domains. (An obvious example is planning with non-replenishable resources, whose consumption is completely ignored within the relaxation.) It has thus been an active area of research to design heuristics 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 2013). We build on Katz et al.'s (2013) work, that 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. Katz et al.'s work is theoretical. We add the necessary ingredients to make it practical.

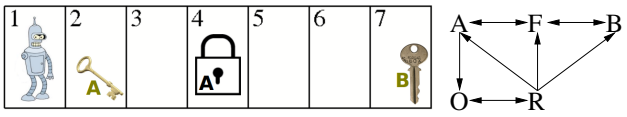
For red-black planning to be useful as the basis of a heuristic function, where a red-black plan will be generated in every search state, red-black plan generation must be tractable. Katz et al. identify two tractable fragments. In one of them, runtime is polynomial but to a very high degree; we do not consider this fragment here. The other fragment imposes a restriction on the causal graph induced by the black variables, and requires the red-black task to be *reversible*: from any reachable state, we must be able to revert all black variables to their initial values. This tractability result is not directly applicable to practice because (A) tractability is shown for plan existence rather than plan generation, and (B) testing red-black reversibility is co-NP-hard.

We identify a refined fragment that fixes both (A) and (B). We impose the same causal graph restriction, but replace reversibility with a stronger notion of *invertibility*, where all black variables are invertible in that every value change can be undone under conditions we know will be true. This sufficient criterion is easily testable, fixing (B). Any delete relaxed plan can with polynomial overhead be turned into a red-black plan, fixing (A).[1] Implementing this in Fast Downward, with some minor refinements and simple techniques for choosing the red variables, we obtain heuristics that can significantly improve on the standard delete relaxation.

Preliminaries

A **finite-domain representation (FDR)** planning task is a tuple $\Pi = \langle V, A, I, G \rangle$. 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*,

[1] The idea of repairing a relaxed plan is related to previous works on generating macros from relaxed plans (Vidal 2004; Baier and Botea 2009). From that perspective, our techniques replace greedy incomplete approximations of real planning with a complete solver for a suitably defined relaxation thereof.



(a) (b)

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

respectively. We identify (partial) assignments with sets of *facts*, i. e., variable-value pairs (v, d). If a has a precondition on v but does not change it, we say that a is *prevailed* by v; the set of all such preconditions is denoted $\mathsf{prevail}(a)$.

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., 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 tuple $\Pi = \langle V, A, I, G \rangle$ 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 ingredient of many planning systems is to exploit 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.

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 $\{(y, x)\}$.

Red-Black Relaxation

Katz et al. view FDR and MFDR as cases in which *all* state variables adopt value-switching and value-accumulating semantics, respectively. They interpolate between these two extremes by what they call *red-black planning*. In what follows, we review their definition, and summarize their results.

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 $v \in \mathcal{V}^{\mathsf{B}}(\mathsf{eff}(a))$ to $\{\mathsf{eff}(a)[v]\}$, and changes the value of $v \in \mathcal{V}^{\mathsf{R}}(\mathsf{eff}(a))$ 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 $v \in \mathcal{V}(G)$.

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 consistent 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.

Katz et al. identify two tractable fragments. One of these requires a fixed number of black variables, with fixed domain size. Runtime is exponential in the product of the do-

main sizes (e. g., exponential in 2^5 if we have 5 binary black variables). This exponent is constant, but much too large to be practical as a procedure to be run in every search state. We thus focus on the other fragment, defined as follows.

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. For a digraph G, denote by **scc-size**(G) the size of the largest strongly connected component. For a set of digraphs $\mathcal{G}$, say that scc-size$(\mathcal{G})$ is **bounded** if there exists a constant k such that scc-size$(G) \leq k$ for all $G \in \mathcal{G}$. By **RB-PlanExist**$(\mathcal{G})$ and **RB-PlanGen**$(\mathcal{G})$, denote the plan existence and plan generation problems restricted to RB tasks whose *black* causal graphs are elements of $\mathcal{G}$ (this imposes no restriction whatsoever on the red variables). Say that Π is **reversible** if, for every state s reachable from I, there exists an action sequence π so that $s[\![\pi]\!][V^{\mathsf{B}}] = I[V^{\mathsf{B}}]$.

Katz et al. prove that, if scc-size$(\mathcal{G})$ is bounded, then RB-PlanExist$(\mathcal{G})$ for reversible RB is polynomial-time solvable. Since fixed-size sets of variables can be replaced with polynomial overhead by single variables, it suffices to show this result for acyclic black causal graphs. As Katz et al. show, red-black plan existence in that setting is equivalent to relaxed plan existence. This tractability result is not directly applicable to practice, for two reasons:

(A) Tractability is shown for plan *existence*, rather than plan *generation* as required in order to compute a heuristic function. It is an open problem whether plan generation is tractable, too; Katz et al. conjecture it is not.

(B) As Katz et al. show, testing red-black reversibility is co-NP-hard. So even if plan generation were polynomial, we would have no efficient way of knowing whether or not we are inside the tractable fragment.

We fix both (A) and (B) by employing a sufficient criterion for reversibility, thus moving to a strictly smaller fragment of RB planning. Throughout the remainder of the paper, we assume that the black causal graph is acyclic (i. e., a DAG).

Invertibility

The basic idea behind our sufficient criterion is to entail reversibility by the ability to "undo" every action application. That is, in principle, not a new idea – similar approaches have been known under the name *invertible actions* for a long time (e. g., (Koehler and Hoffmann 2000)). For every action a, one postulates the existence of a corresponding inverse action a'. That action must be applicable behind a, ensured in FDR by $\mathsf{pre}(a') \subseteq \mathsf{prevail}(a) \cup \mathsf{eff}(a)$; and it must undo a exactly, ensured in FDR by $\mathcal{V}(\mathsf{eff}(a')) = \mathcal{V}(\mathsf{eff}(a)) \subseteq \mathcal{V}(\mathsf{pre}(a))$ and $\mathsf{eff}(a') = \mathsf{pre}(a)[\mathcal{V}(\mathsf{eff}(a))]$. For any reachable state s, we can then revert to the initial state I simply by inverting the path that lead from I to s.

As we now show, our setting allows a much less restrictive definition. What's more, the definition is per-variable, identifying a subset $V_I \subseteq V$ of variables (the invertible ones) that can be painted black in principle. This enables efficient red-black relaxation design: Identify the set V_I, paint all other variables red, keep painting more variables red until there are no more cycles in the black causal graph.

Once we have selected the red variables, every action will affect at most one black variable (otherwise, there would be cycles between the black effect variables). Since red variables accumulate their values anyway, the only effect we need to invert is the black one. This corresponds to inverting a single arc (d, d') in a domain transition graph. Furthermore, both the outside condition ϕ and the outside effect ψ of (d, d') will remain true and can be used as conditions for the inverse arc: For $(u, e) \in \psi$, this is because u must be red. For $(u, e) \in \phi$, if u is red there is nothing to show, and if u is black then (u, e) must be a prevail condition because all the outside effects are red. We obtain the following definition.

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').[2] 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. This can be tested in polynomial time. Furthermore:

Theorem 1 *Any RSE-invertible RB task with acyclic black causal graph is reversible.*

We show that every action application can be undone, i. e., given state s and action a applicable to s, from $s[\![\langle a \rangle]\!]$ we can reach a state s' so that $s'[V^{\mathsf{B}}] = s[V^{\mathsf{B}}]$. If all variables affected by a are red, $s' := s[\![\langle a \rangle]\!]$ does the job. Otherwise, a affects exactly one black variable v. Let (d, d') be the arc in $DTG_{\Pi}(v)$ taken by a in s, let (d', d) be the inverse arc, and let a' be the action that induces (d', d). Then a' is applicable in $s[\![\langle a \rangle]\!]$: Using the notations from above, $\mathsf{pre}(a') \subseteq \phi' \cup \{(v, d')\}$, where $\phi' \subseteq \phi \cup \psi$. As discussed above, both ϕ and ψ are true in $s[\![\langle a \rangle]\!]$. Clearly, $s' := s[\![\langle a, a' \rangle]\!]$ has the required property. This concludes the proof of Theorem 1.

In the example, variables R and F are RSE-invertible. For R, arcs $(i, i+1)$ and $(i+1, i)$ where $\{i, i+1\} \cap \{4\} = \emptyset$ have empty outside conditions so are trivially RSE-invertible; the other arcs all have outside condition $\{(O, 1)\}$ so are RSE-invertible, too. For F, arcs $(1, 0)$ induced by *take*(x, y) are inverted by arcs $(0, 1)$ induced by *drop*(x, y): $\phi' = \{(R, x), (y, R)\}$ is contained in $\phi = \{(R, x), (y, x)\} \cup \psi = \{(y, R)\}$; similarly vice versa.[3]

The outside condition of $DTG_{\Pi}(R)$ arcs $(4, 3)$ and $(4, 5)$ is non-standard in the sense that this condition is not explicitly specified in the IPC version of the GRID domain: instead, the condition is an invariant based on the implicit assumption that the robot is initially in an open position. We have chosen this example to illustrate that, like previous similar notions, RSE-invertibility can be subject to modeling details. It would be an option to use invariance analysis (e. g., (Gerevini and Schubert 1998; Fox and Long 1998; Rintanen 2000; Helmert 2009)) to detect implicit preconditions, but we have not done so for the moment.

[2] This generalizes the earlier definition for actions: v being the black effect variable of a and assuming for simplicity that every effect variable is constrained by a precondition, the requirement is $\mathsf{pre}(a') \subseteq \mathsf{pre}(a) \cup \mathsf{eff}(a)$ and $\mathsf{eff}(a')[v] = \mathsf{pre}(a)[v]$.

[3] Hoffmann (2011) defines a more restricted notion of invertibility (in a different context), requiring $\phi' \subseteq \phi$ rather than $\phi' \subseteq \phi \cup \psi$. Note that, according to this definition, F is not invertible.

Tractable Red-Black Plan Generation

Theorem 1 and Katz et al.'s Theorem 9 immediately imply:

Corollary 1 *Any RSE-invertible RB task* Π *with acyclic black causal graph is solvable if and only if* Π^+ *is.*

In other words, existence of a relaxed plan implies existence of a red-black plan. But how to find that plan? Recall that Katz et al. conjecture there is no polynomial-time "how to" when relying on reversibility. Matters lighten up when relying on RSE-invertibility instead:[4]

Theorem 2 *Plan generation for RSE-invertible RB with acyclic black causal graphs is polynomial-time.*

We show that a relaxed plan can with polynomial overhead be turned into a red-black plan. Figure 2 provides pseudo-code; we use its notations in what follows. The idea is to insert plan fragments achieving the black precondition of the next action, when needed. Likewise for the goal.

Consider an iteration i of the main loop. Any red preconditions of a_i are true in the current state $I[\![\pi]\!]$ because π includes the relaxed plan actions $a_1,\dots,a_{i-1}$ processed so far. Unsatisfied black preconditions g are tackled by $\textsc{Achieve}(\pi, g)$, 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_i is applicable in $I[\![\pi \circ \pi^{\mathsf{B}}]\!]$. Iterating the argument shows the claim.

We next prove that (i) Π^{B} is well-defined, that (ii) all its domain transition graphs are strongly connected, and that (iii) any plan π^{B} for Π^{B} is, in our RB task Π, applicable in the current state $I[\![\pi]\!]$. This suffices because plan generation for FDR with acyclic causal graphs and strongly connected DTGs is tractable: Every such task is solvable, and a plan in a succinct representation can be generated in polynomial time. This follows from Theorem 23 of Chen and Gimenez (2008) with Observation 7 of Helmert (2006). (The succinct representation uses recursive macros for value pairs within DTGs; it is needed as plans may be exponentially long.)

For (i), we need to 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^{B} because by construction these are facts made true by the relaxed plan actions $a_1,\dots,a_{i-1}$ already processed.

For (ii), observe that all values in $DTG_{\Pi^{\mathsf{B}}}(v)$ 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 $DTG_{\Pi^{\mathsf{B}}}(v)$. Say $v \in V^{\mathsf{B}}$, and (d,d') is an arc in $DTG_{\Pi^{\mathsf{B}}}(v)$ induced by a^{B} where $a \in \pi$. Because (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, $\mathsf{pre}(a) \cup \mathsf{eff}(a) \subseteq F$, we get $\mathsf{pre}(a') \subseteq F$. Since a' can have only one black effect, $\mathsf{eff}(a')[V^{\mathsf{B}}] = \{(v,d)\}$ which is contained in F. Thus $a'^{\mathsf{B}} \in A^{\mathsf{B}}$, and (d',d) is an arc in $DTG_{\Pi^{\mathsf{B}}}(v)$ as desired.

[4]This result does *not* hold for the weaker requirement of bounded-size strongly connected components, as replacing fixed-size sets of variables by single variables may lose invertibility.

Algorithm : $\textsc{UnRelax}(\Pi, \pi^+)$

main
// $\Pi = \langle V^{\mathsf{B}}, V^{\mathsf{R}}, A, I, G\rangle$ *and* $\pi^+ = \langle a_1,\dots,a_n\rangle$
$\pi \leftarrow \langle a_1\rangle$
for $i = 2$ **to** n
  do **if** $\mathsf{pre}(a_i)[V^{\mathsf{B}}] \not\subseteq I[\![\pi]\!]$
    then $\pi^{\mathsf{B}} \leftarrow \textsc{Achieve}(\mathsf{pre}(a_i)[V^{\mathsf{B}}])$; $\pi \leftarrow \pi \circ \pi^{\mathsf{B}}$
    $\pi \leftarrow \pi \circ \langle a_i\rangle$
if $G[V^{\mathsf{B}}] \not\subseteq I[\![\pi]\!]$
  then $\pi^{\mathsf{B}} \leftarrow \textsc{Achieve}(G[V^{\mathsf{B}}])$; $\pi \leftarrow \pi \circ \pi^{\mathsf{B}}$

procedure $\textsc{Achieve}(\pi, g)$
$F \leftarrow I \cup \bigcup_{a\in\pi} \mathsf{eff}(a)$
for $v \in V^{\mathsf{B}}$
  do $\mathcal{D}^{\mathsf{B}}(v) \leftarrow \{d \mid d \in \mathcal{D}(v), (v,d) \in F\}$
$I^{\mathsf{B}} \leftarrow I[\![\pi]\!][V^{\mathsf{B}}]$
$G^{\mathsf{B}} \leftarrow g$
$A^{\mathsf{B}} \leftarrow \{a^{\mathsf{B}} \mid a \in A, a^{\mathsf{B}} = \langle \mathsf{pre}(a)[V^{\mathsf{B}}], \mathsf{eff}(a)[V^{\mathsf{B}}]\rangle,$
  $\mathsf{pre}(a) \subseteq F, \mathsf{eff}(a)[V^{\mathsf{B}}] \subseteq F\}$
$\Pi^{\mathsf{B}} \leftarrow \langle V^{\mathsf{B}}, A^{\mathsf{B}}, I^{\mathsf{B}}, G^{\mathsf{B}}\rangle$
$\langle a'^{\mathsf{B}}_1,\dots,a'^{\mathsf{B}}_k\rangle \leftarrow$ an FDR plan for Π^{B}
return $\langle a'_1,\dots,a'_k\rangle$

Figure 2: Algorithm used in the proof of Theorem 2.

Finally, (iii) holds because, with $\mathsf{pre}(a) \subseteq F$ for all actions where $a^{\mathsf{B}} \in A^{\mathsf{B}}$, 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}. This concludes the proof of Theorem 2.

To illustrate, consider again our running example. For $V^{\mathsf{B}} = \{R, F\}$, the example is in our tractable fragment (Figure 1 (b); R and F are RSE-invertible). Say the relaxed plan π^+ is: *move*$(1,2)$, *take*$(2,A)$, *move*$(2,3)$, *open*$(3,4,A)$, *move*$(3,4)$, *move*$(4,5)$, *move*$(5,6)$, *move*$(6,7)$, *take*$(7,B)$, *drop*$(1,B)$. In $\textsc{UnRelax}(\Pi, \pi^+)$, there will be no calls to $\textsc{Achieve}(\pi, g)$ until a_i=*take*$(7,B)$, for which $\textsc{Achieve}(\pi, g)$ constructs Π' with goal $\{(R,7),(F,1)\}$, yielding $\pi' = \langle$*drop*$(7,A)\rangle$ which is added to π. The next iteration calls $\textsc{Achieve}(\pi, g)$ for the precondition of *drop*$(1,B)$, yielding π' moving R back to 1. The algorithm then stops with π being a real plan and yielding the perfect heuristic 17.

The $\textsc{UnRelax}$ algorithm we just presented is feasible in theory. But solving one of the planning tasks Π^{B} roughly corresponds to a call of the causal graph heuristic (Helmert 2006), and there will typically be many such calls for every red-black plan. So, *will the benefit outweigh the computational overhead in practice?* We leave that question open for now, concentrating instead on a much simpler RB fragment:

Corollary 2 *Plan generation for RSE-invertible RB where the black causal graphs contain no arcs is polynomial-time.*

To reach this fragment, "empty" black causal graphs, we keep painting variables red until one of the incident vertices of every causal graph arc is red. During $\textsc{UnRelax}$, solving Π^{B} then consists in finding, for each black v whose value d is not the required one d', a path in $DTG_{\Pi}(v)$ from d to d' all of whose (red) outside conditions are already true. Our

implementation does so using Dijkstra's algorithm. (Note that red-black plans are polynomial-length here.)

It may appear disappointing to focus on empty black causal graphs, given the much more powerful DAG case identified by Theorem 2. Note, however, that the "empty" case is far from trivial: The black variables do not interact directly, but *do* interact via the red variables.[5] Furthermore, as we shall see, even that case poses major challenges, that need to be addressed before proceeding to more complex constructions: The approach is prone to over-estimation.

Implementation

Over-estimation is incurred by following the decisions of a relaxed plan. Say the relaxed plan in our example starts with *move*$(1,2)$, *move*$(2,3)$, *take*$(2,A)$, *open*$(3,4,A)$. Then we need to insert moves from 3 to 2 in front of *take*$(2,A)$, and another move back from 2 to 3 in front of *open*$(3,4,A)$. Similar phenomena occur massively in domains like LOGISTICS, where a relaxed plan may schedule all moves before the loads/unloads: We will uselessly turn the relaxed moves into an actual path, then do the same moves all over again when the loads/unloads come around. To ameliorate this, we employ a simple relaxed plan reordering step before calling UNRELAX. We *forward*, i. e., move as close as possible to the start of the relaxed plan, all actions without black effects.[6] The rationale is that these actions will not interfere with the remainder of the relaxed plan. We denote this technique with **F** for "forward" and omit the "**F**" when not using it. **F** does help to reduce over-estimation, but does not get rid of it. Over-estimation remains the most important weakness of our heuristic; we get back to this later.

A simple optimization is to test, 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.

Finally, we need a technique to choose the red variables. As earlier hinted, 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 of incident arcs to black variables; break ties by smaller domain size.
- **C**: Select v with the minimal number 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-**: Like **C** but with the *maximal* number of conflicts.

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. **C-** is a sanity test painting the *most* important variables red.

[5]To prove Corollary 2 separately, the same observations (i-iii) are required. The only aspect that trivializes is solving Π^{B}.

[6]To avoid having to check relaxed plan validity every time we want to move a in front of a', we just check whether a' achieves a precondition of a, and if so, stop moving a.

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; for space reasons, we present results only for IPC 2002 through to IPC 2011. Since the 2008 domains were run also in 2011, we omit the 2008 benchmarks to avoid a bias on these domains. Our implementation deals with additive action costs, but for the sake of simplicity we consider uniform costs here (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 thus the whole domain for 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 compare to the method proposed by Keyder et al. (2012), that shares with our work the ability to interpolate between monotonic and real planning. Precisely, we use two variants of this heuristic, that we refer 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., across all IPC benchmarks (unpublished; private communication).

We ran 12 variants of our own heuristic, switching **F** and **S** on/off, and running one of **A**, **C**, or **C-**. Table 1 provides an overview pointing out the main observations.

Coverage, and overall coverage especially, is a very crude criterion, but it serves to make basic summary observations. What we see in Table 1 is that our best configuration **AFS** outperforms both FF and Keyder et al. by a reasonable margin. We see that the **F** technique helps a bit, and that **S** is crucial for coverage on these benchmarks (although not for FF, because relaxed plans rarely are real plans). The success of **F** can be traced to improved heuristic accuracy (data omitted). The success of **S** is mainly due to Visitall, where the red-black plan for the initial state always solves the original planning task, so this domain is solved without search (the same happens, by the way, in the Logistics and Miconic domains not shown in the table).[7]

C- performs better than **C**. This is interesting because in **C-** the "most important" variables are red, suggesting that our heuristic becomes better as we make less use of its power. We get back to this below. Note though that the disadvantage of **CFS** is primarily due to Elevators, without

[7]We note that (reported e. g. by Keyder et al.) the FF heuristic covers Visitall when given much more memory. Covering it without search is, of course, better for any memory limit.

		Coverage									Time FF/OWN		Evaluations FF/OWN						H-time OWN/FF	
							A			C	C-	AFS	CFS	AFS			CFS			AF	
	#	Keyd'12	Keyd'13	FF	FF S	FS	S	F	FS	FS	med	med	min	med	max	min	med	max	med	max
Barman	20/20	4	18	19	19	20	19	20	20	20	7.56	122.86	0.80	8.40	433.04	1.44	134.80	513.57	1.24	1.56
Depot	22/22	21	21	18	18	19	19	18	19	19	0.91	0.93	0.15	1.03	46.50	0.15	1.03	46.50	1.06	3.73
Driverlog	20/20	20	20	20	20	20	20	20	20	20	1.00	1.00	0.28	0.96	16.00	0.23	1.02	10.51	1.08	2.96
Elevators	20/20	19	18	20	20	20	20	20	8	20	0.92	0.08	0.54	1.00	5.84	0.02	0.11	0.67	1.09	1.39
Floortile	20/20	20	20	5	5	5	6	5	5	5	0.51	0.46	0.54	0.89	237.06	0.54	0.89	237.06	1.83	7.10
Nomystery	20/20	6	6	10	10	6	5	6	6	6	0.88	0.88	0.01	0.66	2.19	0.01	0.66	2.19	1.03	1.62
Parcprinter	13/20	3	1	13	13	13	9	13	13	13	1.00	1.00	0.01	1.08	1708.22	0.01	1.08	1708.22	1.01	12.84
Pegsol	20/20	19	20	20	20	20	20	20	20	20	1.00	1.00	0.14	1.02	7.35	0.10	1.00	21.83	1.00	3.22
Pipes-notank	40/50	33	33	33	33	33	33	33	33	33	0.91	1.00	0.79	1.02	16.62	0.75	1.03	17.31	1.08	3.36
Pipes-tank	40/50	29	29	29	29	29	31	29	29	29	0.91	0.70	0.14	1.01	70.85	0.13	1.00	70.85	1.33	2.71
Psr-small	50/50	50	50	50	50	50	50	50	50	50	1.00	1.00	1.00	1.03	1.29	1.00	1.03	1.29	1.00	5.20
Rovers	40/40	40	40	40	40	40	40	40	40	40	1.00	1.00	0.45	1.19	11.00	0.58	1.27	11.00	1.15	2.08
Satellite	36/36	34	34	36	36	36	36	35	36	36	1.00	1.00	0.61	1.13	3405.50	0.61	1.20	3405.50	1.25	2.01
Scanalyzer	14/20	14	14	14	14	14	14	14	14	14	0.89	1.00	0.33	1.04	2.08	0.06	1.15	10.14	1.12	2.39
Sokoban	20/20	17	16	19	19	19	19	19	18	19	0.59	0.25	0.10	1.01	109.44	0.01	1.12	109.44	1.92	5.28
Tidybot	20/20	15	12	15	15	13	13	13	17	14	0.69	0.87	1.01	1.05	1.73	0.19	1.08	2.49	1.66	2.44
Tpp	30/30	30	30	30	30	29	29	29	29	29	1.00	1.00	0.05	0.78	29.00	0.05	0.78	29.00	1.04	1.93
Transport	20/20	11	11	10	10	10	10	10	8	9	1.25	1.49	0.36	1.22	9.85	0.21	0.64	2.81	1.12	1.91
Trucks	30/30	14	14	18	18	20	19	20	19	18	1.00	0.54	0.02	1.01	38.56	0.02	0.63	92.54	1.13	6.43
Visitall	20/20	20	20	3	3	20	20	4	20	20	14.40	14.40	15533	18813	409538	15533	18813	409538	2.79	3.12
Woodworking	20/20	20	20	20	20	20	20	20	20	20	0.65	0.66	0.92	1.00	6.00	0.92	1.00	6.00	0.92	1.00
Zenotravel	20/20	20	20	20	20	20	20	20	20	20	1.00	1.00	0.54	1.29	6.11	0.54	1.29	6.11	1.14	1.63
$\sum$	555/588	459	467	462	462	476	472	458	464	474										

Table 1: Selected results on IPC 2002–IPC 2011 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/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-, F, 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. Data for these measures are shown in terms of per-task ratios against FF as indicated. We show features (median, min, max) of that ratio's per-domain distribution on the instances solved by both planners involved.

which it would tie in with **AFS** for first place. Thus the remainder of the table focuses on these two configurations.

A quick glance at the runtime data shows that there are not one, but two, domains where our techniques dramatically beat FF. The median runtime speed-up for **AFS** in Barman is 122.86. Apart from these two domains, the differences in median runtime are small (most notable exception: **AFS** in Elevators) and the advantage is more often on FF's side.

It turns out to be crucial to have a look beyond median performance, which we do for search space size (Evaluations in Table 1). In 14 domains for **AFS**, and in 15 domains for **CFS**, the maximum search space reduction is by 1–3 orders of magnitude! This still is true of 9 (10) domains for **AF** (**CF**). The median (and the geometric mean) does not show this because, in all but Barman and Visitall, the reduction is outweighed by equally large search space *increases* in other instances of the domain.

The heuristic runtime data in Table 1 simply shows that the slow-down typically is small.

Discussion

The red-black relaxed plan heuristics we propose outperform the relaxed plan heuristic consistently in two domains, and outperform it *sometimes* in many domains. On the negative side, in these latter domains they equally often deteriorate performance, and sometimes selecting the "wrong" red variables (**C-**) works better. Both negative observations appear to have one common reason: *unstable heuristic values caused by over-estimation due to arbitrary choices made in the relaxed plans used as input to the* UNRELAX *algorithm.*

In ELEVATORS, e. g., **C** paints the lift position variables black, which seems the right choice because these have to move back and forth in a plan (**C-** tends to paint passenger variables black). But the relaxed plan will typically use actions of the form *move(c,X)* where *c* is the current lift position and *X* are floors that need to be visited. Given this input, whenever the lift needs to go from *Y* to *X*, UNRELAX uses not *move(Y,X)* but *move(Y,c)* followed by *move(c,X)*. Further, even when reordering the relaxed plan using **F**, boarding/leaving actions are not reliably moved up front, causing UNRELAX to uselessly move elevators back and forth without taking care of any passengers. If lift capacity variables are black, every time a passenger boards/leaves, UNRELAX may uselessly board/leave other passengers because relaxed plans are free to choose whichever capacity values. The latter two issues are especially problematic as their impact on red-black plan length is highly dependent on arbitrary per-state choices, with an unpredictable effect on search. It is this per-state arbitrariness that likely leads to the observed dramatic search space size variance in many domains.

In conclusion, red-black relaxed plan heuristics have great potential to reduce search, but we need more stable ways to compute them. One could minimize conflicts in the input relaxed plans (Baier and Botea 2009). Research is needed into red-black planning methods that rely less on relaxed plans, or that do not rely on these at all. We are confident this journey will ultimately lead to a quantum leap in satisficing heuristic search planning.

Acknowledgments. Carmel Domshlak's work was partially supported by ISF grant 1045/12 and the Technion-Microsoft E-Commerce Research Center.

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. 1998. The automatic inference of state invariants in TIM. *Journal of Artificial Intelligence Research* 9:367–421.

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

Gerevini, A., and Schubert, L. 1998. Inferring state-constraints for domain independent planning. In Mostow, J., and Rich, C., eds., *Proceedings of the 15th National Conference of the American Association for Artificial Intelligence (AAAI-98)*, 905–912. Madison, WI, USA: MIT Press.

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.

Helmert, M. 2009. Concise finite-domain representations for PDDL planning tasks. 173:503–535.

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

Hoffmann, J. 2011. Analyzing search topology without running any search: On the connection between causal graphs and h^+. *Journal of Artificial Intelligence Research* 41:155–229.

Katz, M.; Hoffmann, J.; and Domshlak, C. 2013. 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.

Koehler, J., and Hoffmann, J. 2000. On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm. *Journal of Artificial Intelligence Research* 12:338–386.

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

Rintanen, J. 2000. An iterative algorithm for synthesizing invariants. In Kautz, H. A., and Porter, B., eds., *Proceedings of the 17th National Conference of the American Association for Artificial Intelligence (AAAI-00)*, 806–811. Austin, TX, USA: AAAI Press.

Vidal, V. 2004. A lookahead strategy for heuristic search planning. In Koenig, S.; Zilberstein, S.; and Koehler, J., eds., *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, 150–160. Whistler, Canada: Morgan Kaufmann.