Red-Black Planning: A New Tractability Analysis and Heuristic Function

Daniel Gnad and Jörg Hoffmann
Saarland University
Saarbrücken, Germany
\{gnad, hoffmann\}@cs.uni-saarland.de

Abstract

Red-black planning is a recent approach to partial delete relaxation, where red variables take the relaxed semantics (accumulating their values), while black variables take the regular semantics. Practical heuristic functions can be generated from tractable sub-classes of red-black planning. Prior work has identified such sub-classes based on the black causal graph, i.e., the projection of the causal graph onto the black variables. Here, we consider cross-dependencies between black and red variables instead. We show that, if no red variable relies on black preconditions, then red-black plan generation is tractable in the size of the black state space, i.e., the product of the black variables. We employ this insight to devise a new red-black plan heuristic in which variables are painted black starting from the causal graph leaves. We evaluate this heuristic on the planning competition benchmarks. Compared to a standard delete relaxation heuristic, while the increased runtime overhead often is detrimental, in some cases the search space reduction is strong enough to result in improved performance overall.

Introduction

In classical AI planning, we have a set of finite-domain state variables, an initial state, a goal, and actions described in terms of preconditions and effects over the state variables. We need to find a sequence of actions leading from the initial state to a goal state. One prominent way of addressing this is heuristic forward state search, and one major question in doing so is how to generate the heuristic function automatically, i.e., just from the problem description without any further human user input. We are concerned with that question here, in satisficing planning where no guarantee on plan quality needs to be provided. The most prominent class of heuristic functions for satisficing planning are relaxed plan heuristics (e.g., (McDermott 1999; Bonet and Geffner 2001; Hoffmann and Nebel 2001; Gerevini, Saetti, and Serina 2003; Richter and Westphal 2010)).

Relaxed plan heuristics are based on the delete (or monotonic) relaxation, which assumes that state variables accumulate their values, rather than switching between them. Optimal delete-relaxed planning is \textbf{NP}-hard, but satisficing delete-relaxed planning is polynomial-time (Bylander 1994). Given a search state \( s \), relaxed plan heuristics generate a (not necessarily optimal) delete-relaxed plan for \( s \), resulting in an inadmissible heuristic function which tends to be very informative on many planning benchmarks (an explicit analysis has been conducted by Hoffmann (2005)).

Yet, like any heuristic, relaxed plan heuristics also have significant pitfalls. A striking example (see, e.g., (Coles et al. 2008; Nakhost, Hoffmann, and Müller 2012; Coles et al. 2013)) is “resource persistence”, that is, the inability to account for the consumption of non-replenishable resources. As variables never lose any “old” values, the relaxation pretends that resources are never actually consumed. For this and related reasons, the design of heuristics that take some deletes into account has been an active research area from the outset (e.g. (Fox and Long 2001; Gerevini, Saetti, and Serina 2003; Helmert 2004; van den Briel et al. 2007; Helmert and Geffner 2008; Coles et al. 2008; Keyder and Geffner 2008; Baier and Botea 2009; Keyder, Hoffmann, and Haslum 2012; Coles et al. 2013; Keyder, Hoffmann, and Haslum 2014)). We herein continue the most recent approach along these lines, red-black planning as introduced by Katz et al. (2013b).

Red-black planning delete-relaxes only a subset of the state variables, called “red”, which accumulate their values; the remaining variables, called “black”, retain the regular value-switching semantics. The idea is to obtain an inadmissible yet informative heuristic in a manner similar to relaxed plan heuristics, i.e. by generating some (not necessarily optimal) red-black plan for any given search state \( s \). For this to make sense, such red-black plan generation must be sufficiently fast. Therefore, after introducing the red-black planning framework, Katz et al. embarked on a line of work generating red-black plan heuristics based on tractable fragments. These are characterized by properties of the projection of the causal graph – a standard structure capturing state variable dependencies – onto the black variables (Katz, Hoffmann, and Domshlak 2013a; Katz and Hoffmann 2013; Domshlak, Hoffmann, and Katz 2015). Cross-dependencies between black and red variables were not considered at all yet. We fill that gap, approaching “from the other side” in that we analyze only such cross-dependencies. We ignore the structure inside the black part, assuming that there is a single black variable only; in practice, that “single variable” will correspond to the cross-product of the black variables.
Distinguishing between (i) black-precondition-to-red-effect, (ii) red-precondition-to-black-effect, and (iii) mixed-red-black-effect dependencies, and assuming there is a single black variable, we establish that (i) alone governs the borderline between \(P\) and \(NP\): If we allow type (i) dependencies, deciding red-black plan existence is \(NP\)-complete, and if we disallow them, red-black plan generation is exponential-time. Katz et al. also considered the single-black-variable case. Our hardness result strengthens theirs in that it shows only type (i) dependencies are needed. Our tractability result is a major step forward in that it allows to scale the size of the black variable, in contrast to Katz et al.’s algorithm whose runtime is exponential in that parameter. Hence, in contrast to Katz et al.’s algorithm, ours is practical.

It leads us to a new red-black plan heuristic, whose painting strategy draws a “horizontal line” through the causal graph viewed as a DAG of strongly connected components (SCC), with the roots at the top and the leaves at the bottom. The part above the line gets painted red, the part below the line gets painted black, so type (i) dependencies are avoided.

Note that, by design, the black variables must be “close to the causal graph leaves”. This is in contrast with Katz et al.’s red-black plan heuristics, which attempt to paint black the variables “close to the causal graph root”, to account for the to-and-fro of these variables when servicing other variables (e.g., a truck moving around to service packages). Indeed, if the black variables are causal graph leaves, then provably no information is gained over a standard relaxed plan heuristic (Katz, Hoffmann, and Domshlak 2013b). However, in our new heuristic we paint black leaf SCCs, as opposed to leaf variables. As we point out using an illustrative example, this can result in better heuristic estimates than a standard relaxed plan, and even than a red-black plan when painting the causal graph roots black. That said, in the International Planning Competition (IPC) benchmarks, this kind of structure seems to be rare. Our new heuristic often does not yield a search space reduction so its runtime overhead ends up being detrimental. Katz et al.’s heuristic almost universally is detrimental. Katz et al.’s heuristic almost universally is detrimental.

Prior work on tractability in red-black planning (Katz, Hoffmann, and Domshlak 2013b; 2013a; Domshlak, Hoffmann, and Katz 2015) considered (a) the “black causal graph” i.e. the sub-graph induced by the black variables only, and (b) the case of a single black variable. Of these, only (a) was employed for the design of heuristic functions. Herein, we improve upon (b). Method (a) is not of immediate relevance to our technical contribution, but we compare to it empirically, specifically to the most competitive heuristic \(h^{FF}\) as used in the Mercury system that participated in IPC 14 (Katz and Hoffmann 2014). That heuristic exploits the tractable fragment of red-black planning where the black causal graph is acyclic and every black variable is “invertible” in a particular sense. The painting strategy is geared at painting black the “most influential” variables, close to the causal graph roots.

**Example 1** As an illustrative example, we use a simplified version of the IPC benchmark TPP. Consider Figure 1. There is a truck moving along a line \(l_1, \ldots, l_7\) of locations. The truck starts in the middle; the goal is to buy two units of a product, depicted in Figure 1 (a) by the barrels, where one unit is on sale at each extreme end of the road map.

Concretely, say the encoding in FDR is as follows. The state variables are \(T\) with domain \(\{l_1, \ldots, l_7\}\) for the truck position; \(B\) with domain \(\{0, 1, 2\}\) for the amount of product bought already; \(P_1\) with domain \(\{0, 1\}\) for the amount of product still on sale at \(l_1\); and \(P_2\) with domain \(\{0, 1\}\) for the amount of product still on sale at \(l_7\). The initial state is as...
shown in the figure, i.e., $T = l_3$, $B = 0$, $P_1 = 1$, $P_7 = 1$. The goal is $B = 2$. The actions are:

- move$(x,y)$: precondition $\{ T = l_x \}$ and effect $\{ T = l_y \}$, where $x,y \in \{1,\ldots,7\}$ such that $|x-y| = 1$.
- buy$(x,y,z)$: precondition $\{ T = l_x, P_z = 1, B = y \}$ and effect $\{ P_z = 0, B = z \}$, where $x \in \{1,7\}$ and $y,z \in \{0,1,2\}$ such that $z = y + 1$.

The causal graph is shown in Figure 1 (b). Note that the variables pertaining to the product, i.e., $B, P_1, P_7$, form a strongly connected component because of the “buy” actions.

Consider first the painting where all variables are red, i.e., a full delete relaxation. A relaxed plan then ignores the strongly connected component because of the “buy” actions.

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

We focus on the case of a single black variable. This has been previously investigated by Katz et al. (2013b), but scantly only. We will discuss details below; our contribution regards a kind of dependency hitherto ignored, namely cross-dependencies between red and black variables:

Definition 1 Let $\Pi = \langle V^B, V^R, A, I, G \rangle$ be a RB planning task. We say that $v, v' \in V^B \cup V^R$ have different colors if either $v \in V^B$ and $v' \in V^R$ or vice versa. The red-black causal graph $CG_{RB}^{\Pi}$ of $\Pi$ is the digraph with vertices $V$ and those arcs $(v, v')$ from $CG_{RB}^{\Pi}$ where $v$ and $v'$ have different colors. We say that $(v, v')$ is of type:

(i) BtoR if $v \in V^B$, $v' \in V^R$, and there exists an action $a \in A$ such that $(v, v') \in V(\text{pre}(a)) \times V(\text{eff}(a))$.

(ii) RtoB if $v \in V^R$, $v' \in V^B$, and there exists an action $a \in A$ such that $(v, v') \in V(\text{pre}(a)) \times V(\text{eff}(a))$.

(iii) EFF else.

We investigate the complexity of satisficing red-black planning as a function of allowing vs. disallowing each of the types (i) – (iii) of cross-dependencies individually. We completely disregard the inner structure of the black part of $\Pi$, i.e., the subset $V^B$ of black variables may be arbitrary. The underlying assumption is that these variables will be pre-composed into a single black variable. Such “pre-composition” essentially means to build the cross-product of the respective variable domains (Seipp and Helmert 2011). We will refer to that cross-product as the black state space, and state our complexity results relative to the assumption that $|V^B| = 1$, denoting the single black variable with $v^B$.

In other words, our complexity analysis is relative to the size $|D(v^B)|$ of the black state space, as opposed to the size of the input task. From a practical perspective, which we elaborate on in the next section, this makes sense provided the variable painting is chosen so that the black state space is “small”.

Katz et al. (2013b) show in their Theorem 1, henceforth called “KatzP”, that satisficing red-black plan generation is polynomial-time in case $|D(v^B)|$ is fixed, via an algorithm that is exponential only in that parameter. They show in their Theorem 2, henceforth called “KatzNP”, that deciding red-black plan existence is $\text{NP}$-complete if $|D(v^B)|$ is allowed to scale. They do not investigate any structural criteria distinguishing sub-classes of the single black variable case. We close that gap here, considering the dependency types (i) – (iii) of Definition 1. The major benefit of doing so will be a polynomial-time algorithm for scaling $|D(v^B)|$.

Switching each of (i) – (iii) on or off individually yields a lattice of eight sub-classes of red-black planning. It turns out that, as far as the complexity of satisficing red-black plan generation is concerned, this lattice collapses into just two classes, characterized by the presence or absence of dependencies (i): If arcs of type BtoR are allowed, then the problem is $\text{NP}$-complete even if arcs of types RtoB and EFF are disallowed. If arcs of type BtoR are disallowed, then the problem is polynomial-time even if arcs of types RtoB and EFF are allowed. We start with the negative result:

Theorem 1 Deciding red-black plan existence for RB planning tasks with a single black variable, and without $CG_{RB}^{\Pi}$ arcs of types RtoB and EFF, is $\text{NP}$-complete.

Proof: Membership follows from KatzNP. (Plan length with a single black variable is polynomially bounded, so this holds by guess-and-check.)

Figure 2: Illustration of the black variable $v^B$ in the SAT reduction in the proof of Theorem 1.

We prove hardness by a reduction from SAT. Consider a CNF formula $\phi$ with propositional variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$. Our RB planning task has $m$ Boolean red variables $v^R_i$, and the single black variable $v^B$ has domain $\{d_0, \ldots, d_n \} \cup \{x_i, \neg x_i \mid 1 \leq i \leq n \}$. In the initial
Algorithm NoBtoR-Planning:
\[ R := I \cup \text{RedFixedPoint}(A^R) \]
if \[ G[V^R] \subseteq R \] and BlackReachable(R, I[v^B], G[v^B]) then
return “solvable” /\* case (a) */
endif
\[ R := I \cup \text{RedFixedPoint}(A^R \cup A^{RB}) \]
if \[ G[V^R] \subseteq R \] then
for \( a \in A^{RB} \) s.t. \( \text{pre}(a) \not\subseteq R \) do
if BlackReachable(R, \( \text{eff}(a)[v^B] \), G[v^B]) then
return “solvable” /\* case (b) */
endif
endfor
endif
return “unsolvable”

Figure 3: Algorithm used in the proof of Theorem 2.

state, all \( v^R \) are set to false and \( v^B \) has value \( d_0 \). The goal is for all \( v^B \) to be set to true. The actions moving \( v^B \) have preconditions and effects only on \( v^B \), and are such that we can move as shown in Figure 2, i.e., for \( 1 \leq i \leq n \) from \( d_{i-1} \) to \( x_i \); from \( d_{i-1} \) to \( \neg x_i \); from \( x_i \) to \( d_i \); and from \( \neg x_i \) to \( d_i \). For each literal \( l \in \mathcal{E} \) there is an action allowing to set \( v^R \) to true provided \( v^B \) has the correct value, i.e., for \( l = x_i \), the precondition is \( v^B = x_i \), and for \( l = \neg x_i \) the precondition is \( v^B = \neg x_i \). This construction does not incur any RtoB or EFF dependencies. The paths \( v^B \) can take correspond exactly to all possible truth value assignments. We can achieve the red goal iff one of these paths visits at least one literal from every clause, which is the case iff \( \phi \) is satisfiable.

The hardness part of KatzNP relies on EFF dependencies. Theorem 1 strengthens this in showing that these dependencies are not actually required for hardness.

Theorem 2 Satisficing plan generation for RB planning tasks with a single black variable, and without CG\(_{RB}^\Pi\) arcs of type BtoR, is polynomial-time.

Proof: Let \( \Pi = \langle (v^B), V^R, A, I, G \rangle \) as specified. We can partition \( A \) into the following subsets:

- \( A^B := \{ a \in A \mid V(\text{eff}(a)) \cap V^B = \{ v^B \}, V(\text{eff}(a)) \cap V^R = \emptyset \} \) are the actions affecting only the black variable.

- \( A^R := \{ a \in A \mid V(\text{eff}(a)) \cap V^B = \emptyset, V(\text{eff}(a)) \cap V^R \neq \emptyset \} \) are the actions affecting only red variables. As there are no CG\(_{RB}^\Pi\) arcs of type BtoR, the actions in \( A^R \) have no black preconditions.

- \( A^{RB} := \{ a \in A \mid V(\text{eff}(a)) \cap V^B = \{ v^B \}, V(\text{eff}(a)) \cap V^R \neq \emptyset \} \) are the actions affecting both red variables and the black variable. As there are no CG\(_{RB}^\Pi\) arcs of type BtoR, the actions in \( A^{RB} \) have no black preconditions.

Consider Figure 3. By RedFixedPoint(A') for a subset \( A' \subseteq A \) of actions without black preconditions, we mean all red facts reachable using only \( A' \), ignoring any black effects. This can be computed by building a relaxed planning graph over \( A' \). By BlackReachable(R, d, d') we mean the question whether there exists an \( A^B \) path moving \( v^B \) from \( d \) to \( d' \), using only red preconditions from \( R \).

Clearly, NoBtoR-Planning runs in polynomial time. If it returns “solvable”, we can construct a plan \( \pi^{RB} \) for \( \Pi \) as follows. In case (a), we obtain \( \pi^{RB} \) by any sequence of \( A^R \) actions establishing RedFixedPoint(A') (there are neither black preconditions nor black effects), and attaching a sequence of \( A^B \) actions leading from \( I[v^B] \) to \( G[v^B] \). In case (b), we obtain \( \pi^{RB} \) by: any sequence of \( A^R \) actions establishing RedFixedPoint(A' \cup A^{RB}) (there are no black preconditions); attaching the \( A^R \) action \( a \) successfully in the for-loop (which is applicable due to \( \text{pre}(a) \subseteq R \) and \( V(\text{pre}(a)) \cap V^B = \emptyset \)); and attaching a sequence of \( A^B \) actions leading from \( \text{eff}(a)[v^B] \) to \( G[v^B] \). Note that, after RedFixedPoint(A' \cup A^{RB}), only a single \( A^{RB} \) action \( a \) is necessary, enabling the black value \( \text{eff}(a)[v^B] \) from which the black goal is \( A^B \)-reachable.

If there is a plan \( \pi^{RB} \) for \( \Pi \), then NoBtoR-Planning returns “solvable”. First, if \( \pi^{RB} \) does not use any \( A^R \) action, i.e. \( \pi^{RB} \) consists entirely of \( A^B \) and \( A^B \) actions, then case (a) will apply because RedFixedPoint(A') contains all we can do with the former, and BlackReachable(R, I[v^B], G[v^B]) examines all we can do with the latter. Second, say \( \pi^{RB} \) does use at least one \( A^R \) action. RedFixedPoint(A' \cup A^{RB}) contains all red facts that can be achieved in \( \Pi \), so in particular (* RedFixedPoint(A' \cup A^{RB}) contains all red facts true along \( \pi^{RB} \). Let \( a \) be the last \( A^R \) action applied in \( \pi^{RB} \). Then \( \pi^{RB} \) contains a path from \( \text{eff}(a)[v^B] \) to \( G[v^B] \) behind \( a \). With (*), \( \text{pre}(a) \subseteq R \) and BlackReachable(R, \( \text{eff}(a)[v^B] \), G[v^B]) succeeds, so case (b) will apply.

In other words, if (a) no \( A^R \) action is needed to solve \( \Pi \), then we simply execute a relaxed planning fixed point prior to moving \( v^B \). If (b) such an action is needed, then we mix \( A^R \) with the fully-red ones in the relaxed planning fixed point, which works because, having no black preconditions, once an \( A^R \) action has become applicable, it remains applicable. Note that the case distinction (a) vs. (b) is needed: When making use of the “large” fixed point RedFixedPoint(A' \cup A^{RB}), there is no guarantee we can get \( v^B \) back into its initial value afterwards.

Example 2 Consider again our illustrative example (cf. Figure 1), painting \( T \) red and painting all other variables black. Then \( v^B \) corresponds to the cross-product of variables \( B, P_1, \) and \( P_2 \); \( A^R \) contains the “buy” actions, \( A^R \) contains the “move” actions, and \( A^{RB} \) is empty.

The call to RedFixedPoint(A') in Figure 3 results in \( R \) containing all track positions, \( R = \{ T = l_1, \ldots , T = l_7 \} \). The call to BlackReachable(R, I[v^B], G[v^B]) then succeeds as, given that we have both track preconditions \( T = l_1 \) and \( T = l_7 \) required for the “buy” actions, indeed the black goal \( B = 2 \) is reachable. The red-black plan extracted will contain a sequence of moves reaching all of \( \{ T = l_1, \ldots , T = l_7 \} \), followed by a sequence of two “buy” actions leading from \( I[v^B] = \{ B = 0, P_1 = 1, P_2 = 1 \} \) to \( G[v^B] = \{ B = 2 \} \).

Theorem 2 is a substantial improvement over KatzP in terms of the scaling behavior in \( |D(v^B)| \). KatzP is based
on an algorithm with runtime exponential in $|D(v^B)|$. Our NoBtoR-Planning has low-order polynomial runtime in that parameter, in fact all we need to do is find paths in a graph of size $|D(v^B)|$. This dramatic complexity reduction is obtained at the price of disallowing BtoR dependencies.

**Heuristic Function**

Assume an input FDR planning task $\Pi$. As indicated, we will choose a painting (a subset $V^R$ of red variables) so that BtoR dependencies do not exist, and for each search state $s$ generate a heuristic value by running NoBtoR-Planning with $s$ as the initial state. We describe our painting strategy in the next section. Some words are in order regarding the heuristic function itself, which diverges from our previous theoretical discussion – Figure 3 and Theorem 2 – in several important aspects.

While the previous section assumed that the entire black state space is pre-composed into a single black variable $v^B$, that assumption was only made for convenience. In practice there is no need for such pre-composition. We instead run NoBtoR-Planning with the BlackReachable$(R, d, d')$ calls implemented as a forward state space search within the projection onto the black variables, using only those black-affecting actions whose red preconditions are contained in the current set of red facts $R$. This is straightforward, and avoids having to generate the entire black state space up front – instead, we will only generate those parts actually needed during red-black plan generation as requested by the surrounding search. Still, of course for this to be feasible we need to keep the size of the black state space at bay.

That said, actually what we need to keep at bay is not the black state space itself, but its weakly connected components. As the red variables are taken out of this part of the problem, chances are that the remaining part will contain separate components.

**Example 3** In our running example, say there are several different kinds of products, i.e. the truck needs to buy a goal amount of several products. (This is indeed the case in the TPP benchmark suite as used in the IPC.) The state variables for each product then form an SCC like the variables $B, P_1, P_7$ in Figure 1 (b), mutually separated from each other by taking out (painting red) the central variable $T$.

We can decompose the black state space, handling each connected component of variables $V^B \subseteq V^B$ separately. When calling BlackReachable$(R, d, d')$, we do not call a single state space search within the projection onto $V^B$, but call one state space search within the projection onto $V^B_c$, for every component $V^B_c$. The overall search is successful if all its components are, and in that case the overall solution path results from simple concatenation.

We finally employ several simple optimizations: black state space results caching, stop search, and optimized red-black plan extraction. The first of these is important as the heuristic function will be called on the same black state space many times during search, and within each call there may be several questions about paths from $d$ to $d'$ through that state space. The same pairs $d$ and $d'$ may reappear many times in the calls to BlackReachable$(R, d, d')$, so we can avoid duplicate effort simply by caching these results. Precisely, our cache consists of pairs $(d, d')$ along with a black path $\pi(d, d')$ from $d$ to $d'$. (In preliminary experiments, caching the actual triples $(R, d, d')$ led to high memory consumption.) Whenever a call to BlackReachable$(R, d, d')$ is made, we check whether $(d, d')$ is in the cache, and if so check whether $\pi(d, d')$ works given $R$, i.e., contains only actions whose red preconditions are contained in $R$. If that is not so, or if $(d, d')$ is not in the cache at all yet, we run the (decomposed) state space search, and in case of success add its result to the cache.

Stop search is the same as already used in (and found to be important in) Katz et al.’s previous work on red-black plan heuristics. If the red-black plan $\pi_{RB}$ generated for a search state $s$ is actually executable in the original FDR input planning task, then we terminate search immediately and output the path to $s$, followed by $\pi_{RB}$, as the solution.

Finally, the red-black plans $\pi_{RB}$ described in the proof of Theorem 2 are of course highly redundant in that they execute the entire red fixed points, as opposed to establishing only those red facts $R^g \subseteq R$ required by the red goal $G[V^R]$, and required as red preconditions on the solution black path found by BlackReachable$(R, d, d')$. We address this straightforwardly following the usual relaxed planning approach. The forward red fixed point phase is followed by a backward red plan extraction phase, in which we select supporters for $R^g$ and the red subgoals it generates.

**Painting Strategy**

Given an input FDR planning task $\Pi$, we need to choose our painting $V^R$ such that the red-black causal graph $CG_{RB}^{\Pi}$ has no BtoR dependencies. A convenient view for doing so is to perceive the causal graph $CG_{\Pi}$ as a DAG $D$ of SCCs in which the root SCCs are at the top and the leaf SCCs at the bottom: Our task is then equivalent to drawing a “horizontal line” anywhere through $D$, painting the top part red, and painting the bottom part black. We say that such a painting is non-trivial if the bottom part is non-empty.

**Example 4** In our running example, the only non-trivial painting is the one illustrated in Figure 4.

![Figure 4: The painting in our running example.](image)

If there are several different kinds of products as described in Example 3, then the state variables for each product form a separate component in the bottom part. If there are several trucks, then the “horizontal line” may put any non-empty subset of trucks into the top part.
1. Set the candidates for inclusion to be all leaf SCCs $V \subseteq V$ in $D$.
2. Select a $V_i$ where $\prod_{v \in V'} |D(v)|$ is minimal.
3. Set $V' := V^B \cup V_i$ and find the weakly connected components $V_c^B \subseteq V'$.
4. If $\sum_{v \in V^B} |D(v)| \leq N$, set $V^B := V'$, remove $V_i$ from $D$, and iterate; else, terminate.

Example 5 In our running example, this strategy will result in exactly the painting displayed in Figure 4, provided that $N$ is chosen large enough to accommodate the variable subset $\{B, P_1, P_2\}$, but not large enough to accommodate the entire set of variables.

If there are several different kinds of products, as in the IPC TPP domain, then $N$ does not have to be large to accommodate all products (as each is a separate component), but would have to be huge to accommodate any truck (which would reconnect all these components). Hence, for a broad range of settings of $N$, we end up painting the products black and the trucks red, as desired.

Note that our painting strategy may terminate with the trivial painting ($V^B = \emptyset$), namely if even the smallest candidate $V_i$ breaks the size bound $N$. This will happen, in particular, on all input tasks $\Pi$ whose causal graph is a single SCC, unless $N$ is large enough to accommodate the entire state space. Therefore, in practice, we exclude input tasks whose causal graph is strongly connected.

Experiments
Our techniques are implemented in Fast Downward (FD) (Helmert 2006). For our painting strategy, we experiment with the size bounds $N = \{1k, 10k, 100k, 1m, 10m\}$ (“m” meaning “million”). We run all IPC STRIPS benchmarks, precisely their satisficing-planning test suites, where we obtain non-trivial paintings. This excludes domains whose causal graphs are strongly connected, and it excludes domains where even the smallest leaf SCCs break our size bounds. It turns out that, given this, only 9 benchmark domains qualify, 3 of which have been used in two IPC editions so that we end up with 12 test suites.

As our contribution consists in a new heuristic function, we fix the search algorithm, namely FD’s greedy best-first search with lazy evaluation, and evaluate the heuristic function against its closest relatives. Foremost, we compare to the standard relaxed plan heuristic $h^{FF}$, which we set out to improve upon. More specifically, we compare to two implementations of $h^{FF}$: the one from the FD distribution, and our own heuristic with size bound $N = 0$. The former is more “standard”, but differs from our heuristic even in the case $N = 0$ because these are separate implementations that do not coincide exactly in terms of tie breaking. As we shall see, this seemingly small difference can significantly affect performance. To obtain a more precise picture of which differences are due to the black variables rather than other details, we use $N = 0$, i.e. “our own” $h^{FF}$ implementation, as the main baseline. We also run $h^{Mercury}$, as a representative of the state of the art in alternate red-black plan heuristics.

A few words are in order regarding preferred operators. As was previously observed by Katz et al., $h^{Mercury}$ yields best performance when using the standard preferred operators extracted from $h^{FF}$. The latter is computed as part of computing $h^{FF}$ anyhow, and the standard preferred operators tend to work better than variants Katz et al. tried trying to exploit the handling of black variables in $h^{Mercury}$. For our own heuristics, we made a similar observation, in that we experimented with variants of preferred operators specific to these, but found that using the standard preferred operators from $h^{FF}$ gave better results. This is true despite the fact that our heuristics do not compute $h^{FF}$ as part of the process. The preferred operators are obtained by a separate call to
FD’s standard implementation of $h^{FF}$, on every search state. Hence, in what follows, all heuristics reported use the exact same method to generate preferred operators.

Table 1 shows coverage results. Observe first that, in terms of this most basic performance parameter, $h^{Mercury}$ dominates all other heuristics, across all domains and regardless whether or not preferred operators are being used. Recall here that, in contrast to our heuristics which paint black the variables “close to the causal graph leaves”, $h^{Mercury}$ uses paintings that paint black the variables “close to the causal graph roots”. Although in principle the former kind of painting can be of advantage as illustrated in Example 1, as previously indicated the latter kind of painting tends to work better on the IPC benchmarks. We provide a per-instance comparison of $h^{Mercury}$ against our heuristics, in Rovers and TPP which turn out to be the most interesting domains for these heuristics, further below (Table 3). For now, let’s focus on the comparison to the baseline, $h^{FF}$.

Note first the influence of tie breaking: Without preferred operators, $N = 0$ has a dramatic advantage over $h^{FF}$ in ParcPrinter, and smaller but significant disadvantages in Logistics98, Pathways, Satellite, and TPP. With preferred operators, the coverage differences get smoothed out, because with the pruning the instances become much easier to solve so the performance differences due to the different heuristics do not affect coverage as much anymore. The upshot is that only the advantage in ParcPrinter, but none of the disadvantages, remain. As these differences have nothing to do with our contribution, we will from now on not discuss $h^{FF}$ as implemented in FD, and instead use the baseline $N = 0$.

Considering coverage as a function of $N$, observe that, with preferred operators, there are no changes whatsoever, again because with the pruning the instances become much easier to solve. Without preferred operators, increasing $N$ and thus the black part of our heuristic function affects coverage in Pathways, Rovers, Satellite, TPP, and Woodworking11. With the single exception of Satellite for $N = 1m$, the coverage change relative to the baseline $N = 0$ is positive. However, the extent of the coverage increase is small in almost all cases. We now examine this more closely, considering more fine-grained performance parameters.

Table 2 considers the number of evaluated states during search, and search runtime, in terms of improvement factors i.e. the factor by which evaluations/search time reduce relative to the baseline $N = 0$. As we can see in the top half of the table, the (geo)mean improvement factors are almost consistently greater than 1 (the most notable exception being Pathways), i.e., there typically is an improvement on average (although: see below). The actual search time, on the other hand, almost consistently gets worse, with a very pronounced tendency for the “improvement factor” to be $< 1$, and to decrease as a function of $N$. The exceptions in this regard are Rovers, and especially TPP where, quite contrary to the common trend, the search time improvement factor grows as a function of $N$. This makes sense as Rovers and TPP clearly stand out as the two domains with the highest evaluations improvement factors.

Per-instance data sheds additional light on this. In Logistics, Miconic, ParcPrinter, Pathways, and Zenotravel, almost all search space reductions obtained are on the smallest instances, where $N$ is large enough to accommodate the entire state space and hence, trivially, the number of evaluations is 1. On almost all larger instances of these domains, the search spaces are identical, explaining the bad search time results previously observed. In Satellite and Woodworking, the results are mixed. There are substantial improvements also on some large instances, but the evaluations improvement factor is always smaller than 6, with the single exception of Woodworking08 instance p24 where for $N \geq 10k$ it is 17.23. In contrast, in Rovers the largest evaluations improvement factor is 4612, and in TPP it is 17317.

Table 3 shows per-instance data on Rovers and TPP, where our techniques are most interesting. We also include $h^{Mercury}$ here for a detailed comparison. $N = 1k$ and $N = 100k$ are left out of the table for lack of space, and as these configurations are always dominated by at least one other value of $N$ here. With respect to the behavior against the baseline $N = 0$, clearly in both domains drastic evaluations and search time improvements can be obtained. It should be said though that there is an unfortunate tendency for our red-black heuristics to have advantages in the smaller instances, rather than the larger ones. This is presumably because, in smaller instances (even disregarding the pathological case where the entire state space fits into the black part of our heuristic) we have a better chance to capture complex variable interactions inside the black part, and hence obtain substantially better heuristic values.

Table 3: Evaluations and search time in Rovers and TPP. “E” evaluations, “T” search time. Without preferred operators.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>0</th>
<th>10k</th>
<th>1m</th>
<th>10m</th>
</tr>
</thead>
<tbody>
<tr>
<td>h_Mercury</td>
<td>E</td>
<td>T</td>
<td>E</td>
<td>T</td>
<td>E</td>
</tr>
<tr>
<td>g1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g2</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g3</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g4</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g5</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g6</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g7</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g8</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g9</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g10</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g11</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g12</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g13</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g14</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g15</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g16</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g17</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g18</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g19</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g20</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g21</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g22</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g23</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g24</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g25</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g26</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g27</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g28</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g29</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>g30</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
</tbody>
</table>

With respect to $h_{Mercury}$, the conclusion can only be that the previous approach to red-black plan heuristics – painting variables “close to the root” black, as opposed to painting variables “close to the leaves” black as we do here – works better in practice. There are rare cases where our new heuristics have an advantage, most notably in Rovers p20, p26, p30, and TPP p5–p17, p19, p20. But overall, especially on the largest instances, $h_{Mercury}$ tends to be better. We remark that, with preferred operators switched on, the advantage of $h_{Mercury}$ tends to be even more pronounced because the few cases that are hard for it in Table 3 become easy.

A few words are in order regarding plan quality, by which, since we only consider uniform action costs in the experiments, we mean plan length. Comparing our most informsated configuration, $N = 10m$, to our pure delete relaxed baseline, i.e. our heuristic with $N = 0$, it turns out that the value of $N$ hardly influences the quality of the plans found. Without using preferred operators, the average per-domain gain/loss of one configuration over the other is always less than 3%. The only domain where solution quality differs more significantly is TPP, where the generated plans for $N = 10m$ are 23.3% shorter on average than those with $N = 0$. This reduces to 10% when preferred operators are switched on. In the other domains, not much changes when enabling preferred operators; the average gain/loss per domain is less than 4.4%.

Comparing our $N = 10m$ configuration to $h_{Mercury}$, having preferred operators disabled, the plan quality is only slightly different in most domains (<3.1% gain/loss on average). Results differ much more significantly in Miconic and TPP. In the former, our plans are 25% longer than those found using $h_{Mercury}$; in the latter, our plans are 25% shorter. Enabling preferred operators does not change much, except in Woodworking, where our plans are on average 19.1% (16.5%) shorter in the IPC’08 (IPC’11) instance suits.

Conclusion

Our investigation has brought new insights into the interaction between red and black variables in red-black planning. The practical heuristic function resulting from this can, in principle, improve over standard relaxed plan heuristics as well as known red-black plan heuristics. In practice – as far as captured by IPC benchmarks – unfortunately such improvements are rare. We believe this is a valuable insight for further research on red-black planning. It remains to be seen whether our tractability analysis can be extended and/or exploited in some other, more practically fruitful, way. The most promising option seems to be to seek tractable special cases of black-to-red (BtoR) dependencies, potentially by restrictions onto the DTG (the variable-value transitions) of the black variable weaker than the “invertibility” criterion imposed by Katz et al.

Acknowledgments. We thank Carmel Domshlak for discussions. We thank the anonymous reviewers, whose comments helped to improve the paper. This work was partially supported by the German Research Foundation (DFG), under grant HO 2169/5-1, and by the EU FP7 Programme under grant agreement 295261 (MEALS). Daniel Gnad’s travel to SoCS’15 was partially supported by the AI section of the German Informatics Society (GI).

References


Coles, A.; Fox, M.; Long, D.; and Smith, A. 2008. A hybrid relaxed planning graph/pl heuristic for numeric planning do-


