

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 space 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 still is **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 polynomial-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 performs better. In some cases though, our heuristic does reduce the search space dramatically relative to standard relaxed plans, resulting in improved performance.

Preliminaries

Our approach is placed in the *finite-domain representation (FDR)* framework. To save space, we introduce FDR and its delete relaxation as special cases of red-black planning. A *red-black (RB)* planning task is a tuple $\Pi = \langle V^B, V^R, A, I, G \rangle$. V^B is a set of *black state variables* and V^R is a set of *red state variables*, where $V^B \cap V^R = \emptyset$ and each $v \in V := V^B \cup V^R$ is associated with a finite domain $\mathcal{D}(v)$. The *initial state* I is a complete assignment to V , the *goal* G is a partial assignment to V . Each action a is a pair $\langle \text{pre}(a), \text{eff}(a) \rangle$ of partial assignments to V called *precondition* and *effect*. We often refer to (partial) assignments as sets of *facts*, i. e., variable-value pairs $v = d$. For a partial assignment p , $\mathcal{V}(p)$ denotes the subset of V instantiated by p . For $V' \subseteq \mathcal{V}(p)$, $p[V']$ denotes the value of V' in p .

A state s assigns each $v \in V$ a non-empty subset $s[v] \subseteq \mathcal{D}(v)$, where $|s[v]| = 1$ for all $v \in V^B$. An action a is ap-

plicable in state s if $\text{pre}(a)[v] \in s[v]$ for all $v \in \mathcal{V}(\text{pre}(a))$. Applying a in s changes the value of $v \in \mathcal{V}(\text{eff}(a)) \cap V^B$ to $\{\text{eff}(a)[v]\}$, and changes the value of $v \in \mathcal{V}(\text{eff}(a)) \cap V^R$ to $s[v] \cup \{\text{eff}(a)[v]\}$. The resulting state is denoted $s[a]$. By $s[\langle a_1, \dots, a_k \rangle]$ we denote the state obtained from sequential application of a_1, \dots, a_k . An action sequence $\langle a_1, \dots, a_k \rangle$ is a *plan* if $G[v] \in I[\langle a_1, \dots, a_k \rangle][v]$ for all $v \in \mathcal{V}(G)$.

Π is a *finite-domain representation (FDR)* planning task if $V = V^B$, and is a *monotonic finite-domain representation (MFDR)* planning task if $V = V^R$. Optimal planning for MFDR tasks is **NP**-complete, but satisficing planning is polynomial-time. The latter can be exploited for deriving (inadmissible) *relaxed plan* heuristics, denoted h^{FF} here. Generalizing this to red-black planning, the *red-black relaxation* of an FDR task Π relative to a *variable painting*, i. e. a subset V^R to be painted red, is the RB task $\Pi_{V^R}^{\text{RB}} = \langle V \setminus V^R, V^R, A, I, G \rangle$. A plan for $\Pi_{V^R}^{\text{RB}}$ is a *red-black plan* for Π . Generating optimal red-black plans is **NP**-hard regardless of the painting simply because we always generalize MFDR. The idea is to generate satisficing red-black plans and thus obtain a *red-black plan heuristic* h^{RB} similarly as for h^{FF} . That approach is practical if the variable painting is chosen so that satisficing red-black plan generation is tractable (or sufficiently fast, anyway).

A standard means to identify structure, and therewith tractable fragments, in planning is to capture dependencies between state variables in terms of the *causal graph*. This is a digraph with vertices V . An arc (v, v') is in CG_{Π} if $v \neq v'$ and there exists an action $a \in A$ such that $(v, v') \in [\mathcal{V}(\text{eff}(a)) \cup \mathcal{V}(\text{pre}(a))] \times \mathcal{V}(\text{eff}(a))$.

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^{Mercury} 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, \dots, 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 $\{l_1, \dots, l_7\}$ 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_7 with domain $\{0, 1\}$ for the amount of product still on sale at l_7 . The initial state is as

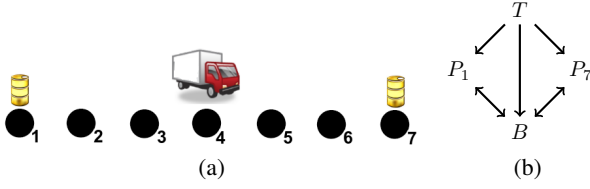


Figure 1: Our running example (a), and its causal graph (b).

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 $x, y \in \{1, \dots, 7\}$ such that $|x - y| = 1$.
- *buy*(x, y, z): precondition $\{T = l_x, P_x = 1, B = y\}$ and effect $\{P_x = 0, B = z\}$, 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 that, after buying the product at one of the two locations l_1 or l_7 , the product is no longer available so we have to move to the other end of the line. Instead, we can buy the product again at the same location, ending up with a relaxed plan of length 5 instead of the 11 steps needed in a real plan.

Exactly the same problem arises in h^{Mercury} : The only “invertible” variable here is T . But if we paint only T black, then the red-black plan still is the same as the fully delete-relaxed plan (the truck does not have to move back and forth anyhow), and we still get the same goal distance estimate 5.

Now say that we paint T red, and paint all other variables black. This is the painting our new heuristic function will use. We can no longer cheat when buying the product, i. e., we do need to buy at each of l_1 and l_7 . Variable T is relaxed so we require 6 moves to reach both these locations, resulting in a red-black plan of length 8.

Tractability Analysis

We focus on the case of a single black variable. This has been previously investigated by Katz et al. (2013b), but scantily 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_{Π}^{RB} of Π is the digraph with vertices V and those arcs (v, v') from CG_{Π} where v and v' have different colors. We say that (v, v') is of type:

- BtoR if $v \in V^B$, $v' \in V^R$, and there exists an action $a \in A$ such that $(v, v') \in \mathcal{V}(\text{pre}(a)) \times \mathcal{V}(\text{eff}(a))$.
- RtoB if $v \in V^R$, $v' \in V^B$, and there exists an action $a \in A$ such that $(v, v') \in \mathcal{V}(\text{pre}(a)) \times \mathcal{V}(\text{eff}(a))$.
- 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 Π , 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 $|\mathcal{D}(v^B)|$ 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 $|\mathcal{D}(v^B)|$ 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 **NP**-complete if $|\mathcal{D}(v^B)|$ 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 $|\mathcal{D}(v^B)|$.

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 **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_{Π}^{RB} arcs of types RtoB and EFF, is **NP**-complete.

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

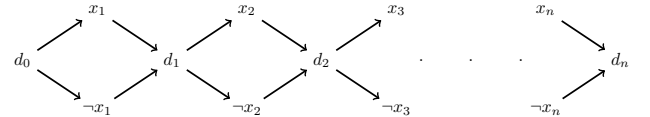


Figure 2: Illustration of the black variable v^B in the SAT reduction in the proof of Theorem 1.

We prove hardness by a reduction from SAT. Consider a CNF formula ϕ with propositional variables x_1, \dots, x_n and clauses c_1, \dots, c_m . Our RB planning task has m Boolean red variables v_i^R , and the single black variable v^B has domain $\{d_0, \dots, d_n\} \cup \{x_i, \neg x_i \mid 1 \leq i \leq n\}$. In the initial

Algorithm NoBtoR-Planning:

```

 $R := I[V^R] \cup \text{RedFixedPoint}(A^R)$ 
if  $G[V^R] \subseteq R$  and  $\text{BlackReachable}(R, I[v^B], G[v^B])$  then
  return “solvable” /* case (a) */
endif
 $R := I[V^R] \cup \text{RedFixedPoint}(A^R \cup A^{RB})$ 
if  $G[V^R] \subseteq R$  then
  for  $a \in A^{RB}$  s.t.  $\text{pre}(a) \subseteq R$  do
    if  $\text{BlackReachable}(R, \text{eff}(a)[v^B], G[v^B])$  then
      return “solvable” /* case (b) */
    endif
  endfor
endif
return “unsolvable”

```

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

state, all v_j^R are set to false and v^B has value d_0 . The goal is for all v_j^R 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 $l \in c_j$ there is an action allowing to set v_j^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 ϕ is satisfiable. \square

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_{Π}^{RB} 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:

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

Consider Figure 3. By $\text{RedFixedPoint}(A')$ 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 $\text{BlackReachable}(R, d, d')$ 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 π^{RB} for Π as follows. In case (a), we obtain π^{RB} by any sequence of A^R actions establishing $\text{RedFixedPoint}(A^R)$ (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 π^{RB} by: any sequence of A^R actions establishing $\text{RedFixedPoint}(A^R \cup A^{RB})$ (there are no black preconditions); attaching the A^{RB} action a successful in the for-loop (which is applicable due to $\text{pre}(a) \subseteq R$ and $\mathcal{V}(\text{pre}(a)) \cap V^B = \emptyset$); and attaching a sequence of A^B actions leading from $\text{eff}(a)[v^B]$ to $G[v^B]$. Note that, after $\text{RedFixedPoint}(A^R \cup A^{RB})$, 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 π^{RB} for Π , then NoBtoR-Planning returns “solvable”. First, if π^{RB} does not use any A^{RB} action, i. e. π^{RB} consists entirely of A^R and A^B actions, then case (a) will apply because $\text{RedFixedPoint}(A^R)$ contains all we can do with the former, and $\text{BlackReachable}(R, I[v^B], G[v^B])$ examines all we can do with the latter. Second, say π^{RB} does use at least one A^{RB} action. $\text{RedFixedPoint}(A^R \cup A^{RB})$ contains all red facts that can be achieved in Π , so in particular (*) $\text{RedFixedPoint}(A^R \cup A^{RB})$ contains all red facts true along π^{RB} . Let a be the last A^{RB} action applied in π^{RB} . Then π^{RB} contains a path from $\text{eff}(a)[v^B]$ to $G[v^B]$ behind a . With (*), $\text{pre}(a) \subseteq R$ and $\text{BlackReachable}(R, \text{eff}(a)[v^B], G[v^B])$ succeeds, so case (b) will apply. \square

In other words, if (a) no A^{RB} action is needed to solve Π , 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^{RB} with the fully-red ones in the relaxed planning fixed point, which works because, having no black preconditions, once an A^{RB} 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 $\text{RedFixedPoint}(A^R \cup A^{RB})$, 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_7 ; A^B contains the “buy” actions, A^R contains the “move” actions, and A^{RB} is empty.*

The call to $\text{RedFixedPoint}(A^R)$ in Figure 3 results in R containing all truck positions, $R = \{T = l_1, \dots, T = l_7\}$. The call to $\text{BlackReachable}(R, I[v^B], G[v^B])$ then succeeds as, given we have both truck 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 $\{T = l_1, \dots, T = l_7\}$, 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 $|\mathcal{D}(v^B)|$. KatzP is based

on an algorithm with runtime exponential in $|\mathcal{D}(v^B)|$. 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 $|\mathcal{D}(v^B)|$. This dramatic complexity reduction is obtained at the price of disallowing BtoR dependencies.

Heuristic Function

Assume an input FDR planning task Π . 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 $\text{BlackReachable}(R, d, d')$ 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 $V_c^B \subseteq V^B$ separately. When calling $\text{BlackReachable}(R, d, d')$, 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_c^B , for every component V_c^B . 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 re-appear many times in the calls to $\text{BlackReachable}(R, d, d')$, 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 $\text{BlackReachable}(R, d, d')$ 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 π^{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 π^{RB} , as the solution.

Finally, the red-black plans π^{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 $\text{BlackReachable}(R, d, d')$. 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 Π , we need to choose our painting V^R such that the red-black causal graph $\text{CG}_{\Pi}^{\text{RB}}$ has no BtoR dependencies. A convenient view for doing so is to perceive the causal graph CG_{Π} 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.

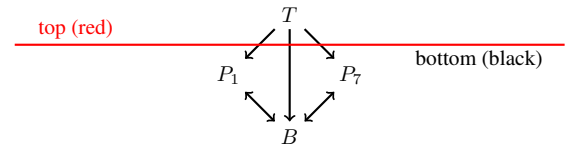


Figure 4: The painting in our running example.

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.

		without preferred operators								with preferred operators							
				$N =$								$N =$					
domain	#	h^{Mercury}	h^{FF}	0	1k	10k	100k	1m	10m	h^{Mercury}	h^{FF}	0	1k	10k	100k	1m	10m
Logistics00	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28
Logistics98	35	35	26	23	23	23	23	23	23	35	35	32	32	32	32	32	32
Miconic	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150
ParcPrinter08	30	30	26	30	30	30	30	30	30	30	26	30	30	30	30	30	30
ParcPrinter11	20	20	12	20	20	20	20	20	20	20	12	20	20	20	20	20	20
Pathways	30	11	11	8	10	10	10	10	10	30	20	23	23	23	23	23	23
Rovers	40	27	23	23	23	24	26	25	24	40	40	40	40	40	40	40	40
Satellite	36	36	30	26	26	26	26	25	26	36	36	35	35	35	35	35	35
TPP	30	23	22	18	18	18	18	18	19	30	30	30	30	30	30	30	30
Woodworking08	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Woodworking11	20	20	19	19	19	20	20	20	20	20	20	20	20	20	20	20	20
Zenotravel	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
Σ	469	430	397	395	397	399	401	399	400	469	447	458	458	458	458	458	458

Table 1: Coverage results. All heuristics are run with FD’s greedy best-first search, single-queue for configurations without preferred operators, double-queue for configurations with preferred operators. The preferred operators are taken from h^{FF} in all cases (see text).

We implemented a simple painting strategy accommodating the above. The strategy has an input parameter N imposing an upper bound on the (conservatively) estimated size of the decomposed black state space. Starting with the DAG D of SCCs over the original causal graph, and with the empty set V^{B} of black variables, iterate the following steps:

1. Set the candidates for inclusion to be all leaf SCCs $V_l \subseteq V$ in D .
2. Select a V_l where $\prod_{v \in V_l} |\mathcal{D}(v)|$ is minimal.
3. Set $V' := V^{\text{B}} \cup V_l$ and find the weakly connected components $V_c^{\text{B}} \subseteq V'$.
4. If $\sum_{V_c^{\text{B}}} \prod_{v \in V_c^{\text{B}}} |\mathcal{D}(v)| \leq N$, set $V^{\text{B}} := V'$, remove V_l 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 N is chosen large enough to accommodate the variable subset $\{B, P_1, P_7\}$, 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 ($V^{\text{B}} = \emptyset$), namely if even the smallest candidate V_l breaks the size bound N . This will happen, in particular, on all input tasks Π 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 $N \in \{1k, 10k, 100k, 1m, 10m\}$ (“ 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^{Mercury} 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

domain	#	1k	10k	100k	1m	10m
evaluations						
Logistics00	28	1.00	1.05	1.43	3.71	3.71
Logistics98	23	1.00	0.98	1.01	1.22	1.35
Miconic	150	1.24	1.41	1.86	2.18	3.12
ParcPrinter08	30	1.07	1.38	1.52	1.52	1.71
ParcPrinter11	20	1.00	1.03	1.08	1.08	1.09
Pathways	8	0.71	0.71	0.71	0.88	0.88
Rovers	19	1.60	1.95	5.10	5.84	5.16
Satellite	25	1.04	1.83	1.61	2.18	2.40
TPP	17	1.83	3.76	5.90	20.89	27.54
Woodworking08	30	1.54	2.06	2.06	2.06	2.06
Woodworking11	19	1.08	1.68	1.76	1.76	1.76
Zenotravel	20	1.14	1.14	0.87	1.13	2.27
search time						
Logistics00	28	1.00	1.00	0.94	0.59	0.59
Logistics98	23	1.00	1.00	0.93	0.80	0.58
Miconic	150	0.73	0.73	0.70	0.63	0.44
ParcPrinter08	30	1.00	1.00	1.00	1.00	0.44
ParcPrinter11	20	1.00	0.95	0.96	0.96	0.23
Pathways	8	0.97	0.97	0.96	0.96	0.93
Rovers	19	1.44	1.74	2.01	1.56	0.87
Satellite	25	0.87	1.07	0.90	0.63	0.24
TPP	17	0.98	1.39	1.78	3.75	4.67
Woodworking08	30	0.94	0.86	0.86	0.86	0.86
Woodworking11	19	0.87	0.65	0.67	0.67	0.67
Zenotravel	20	1.00	1.00	0.95	0.78	0.43

Table 2: Improvement factors relative to $N = 0$. Per-domain geometric mean over the set of instances commonly solved for all values of N . Without preferred operators.

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.

	h^{Mercury}		$N =$							
	E	T	0 E	T	10k E	T	1m E	T	10m E	T
Rovers										
p01	5	0.1	35	0.1	31	0.1	1	0.1	1	0.1
p02	6	0.1	6	0.1	1	0.1	1	0.1	1	0.1
p03	1	0.1	62	0.1	112	0.1	1	0.1	1	0.1
p04	1	0.1	17	0.1	21	0.1	1	0.1	1	0.1
p05	119	0.1	114	0.1	170	0.1	117	0.1	117	0.1
p06	304	0.1	543	0.1	485	0.1	485	0.1	285	0.6
p07	70	0.1	331	0.1	334	0.1	162	1.5	175	7.1
p08	116	0.1	1742189	46.3	451078	15.4	603	0.1	1929	28.2
p09	358	0.1	2773	0.1	1792	0.1	2120	0.1	2120	0.1
p10	578	0.1	441	0.1	441	0.1	244	0.4	244	0.4
p11	1047	0.1	85832	2.7	85787	3.1	85787	3.1	85787	3.1
p12	6	0.1	606	0.1	958	0.1	301	0.3	698	0.4
p13	25037	2.13	1944	0.1	2578	0.1	2882	0.3	2882	0.3
p14	294	0.1	5161720	208.5	1467	0.1	732	0.2	1119	0.8
p15	1035	0.1	—	—	5024	0.3	3520	1.5	4782	3.3
p16	358	0.1	—	—	11895	0.6	11895	0.6	11895	0.6
p17	1139	0.1	—	—	—	—	3340	0.3	3340	0.3
p18	2156	0.19	93372	6.6	47472	3.6	47472	3.6	48620	6.0
p19	180979	22.08	370650	38.3	452905	47.7	470758	47.7	470758	47.3
p20	—	—	1782100	233.1	—	—	1828671	248.3	1828671	248.0
p22	3674	0.63	2478919	339.2	1707251	237.1	1707251	232.4	1707251	236.0
p23	24347	5.77	—	—	—	—	—	—	—	—
p25	1	0.1	2677	0.3	31890	4.5	31248	4.2	44688	5.8
p26	7338129	1418.16	117583	13.7	83724	11.7	7263721	974.8	—	—
p27	61575	13.73	—	—	6737329	1158.0	6737329	1138.6	6737329	1148.3
p28	6346	1.95	1066434	223.8	15321	3.8	18002	4.3	18002	4.3
p29	8409	2.71	1298473	251.4	—	—	—	—	—	—
p30	—	—	5940371	1134.0	333303	108.0	—	—	—	—
p34	60144	38.59	—	—	—	—	—	—	—	—
TPP										
p01	1	0.1	5	0.1	1	0.1	1	0.1	1	0.1
p02	1	0.1	9	0.1	1	0.1	1	0.1	1	0.1
p03	1	0.1	13	0.1	1	0.1	1	0.1	1	0.1
p04	1	0.1	17	0.1	17	0.1	1	0.1	1	0.1
p05	1	0.1	22	0.1	25	0.1	25	0.1	25	0.1
p06	38	0.1	107	0.1	46	0.1	46	0.1	46	0.1
p07	1672	0.1	1756	0.1	68	0.1	68	0.1	68	0.1
p08	2462	0.1	2534	0.1	71	0.1	71	0.1	71	0.1
p09	6753	0.28	2963	0.1	299	0.1	121	0.1	121	0.1
p10	24370	1.24	10712	0.5	1061	0.1	147	0.1	147	0.1
p11	15519	1.3	1610504	99.8	56090	4.0	93	0.1	93	0.1
p12	54852	4.37	1340734	91.3	699377	55.7	109	0.1	109	0.1
p13	38205	3.5	40291	3.5	40291	3.5	472	0.1	475	0.1
p14	57981	7.37	35089	3.6	35089	3.6	552	0.1	555	0.1
p15	52722	7.32	22842	2.4	22842	2.5	70467	8.1	283	0.1
p16	298618	77.47	247304	49.6	247304	48.4	112610	19.6	253727	38.0
p17	2660716	774.05	—	—	—	—	—	—	1611051	671.4
p18	264855	77.25	—	—	—	—	—	—	—	—
p19	1957381	639.02	1710323	509.3	1710323	508.8	1710323	505.6	—	—
p20	—	—	—	—	—	—	—	—	4161990	1272.4
p21	811226	578.23	—	—	—	—	—	—	—	—
p22	652741	372.74	—	—	—	—	—	—	—	—
p23	1329626	902.09	2432228	1362.0	2432228	1365.3	2432228	1367.6	2432228	1370.7
p24	1253699	801.71	—	—	—	—	—	—	—	—

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

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 informed 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 $< 3\%$. 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 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 suites.

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

- 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’09)*, 10–17. 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.
- Coles, A.; Fox, M.; Long, D.; and Smith, A. 2008. A hybrid relaxed planning graph’lp heuristic for numeric planning do-

- mains. In Rintanen, J.; Nebel, B.; Beck, J. C.; and Hansen, E., eds., *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS'08)*, 52–59. AAAI Press.
- Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2013. A hybrid LP-RPG heuristic for modelling numeric resource flows in planning. *Journal of Artificial Intelligence Research* 46:343–412.
- Domshlak, C.; Hoffmann, J.; and Katz, M. 2015. Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence* 221:73–114.
- Fox, M., and Long, D. 2001. Hybrid STAN: Identifying and managing combinatorial optimisation sub-problems in planning. In Nebel, B., ed., *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, 445–450. Seattle, Washington, USA: Morgan Kaufmann.
- 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.
- 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'08)*, 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.
- Hoffmann, J. 2005. Where ‘ignoring delete lists’ works: Local search topology in planning benchmarks. *Journal of Artificial Intelligence Research* 24:685–758.
- Katz, M., and Hoffmann, J. 2013. Red-black relaxed plan heuristics reloaded. In Helmert, M., and Röger, G., eds., *Proceedings of the 6th Annual Symposium on Combinatorial Search (SOCS'13)*, 105–113. AAAI Press.
- Katz, M., and Hoffmann, J. 2014. Mercury planner: Pushing the limits of partial delete relaxation. In *IPC 2014 planner abstracts*, 43–47.
- Katz, M.; Hoffmann, J.; and Domshlak, C. 2013a. Red-black relaxed plan heuristics. In desJardins, M., and Littman, M., eds., *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI'13)*, 489–495. Bellevue, WA, USA: AAAI Press.
- 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'13)*, 126–134. Rome, Italy: AAAI Press.
- 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'12)*, 128–136. AAAI Press.
- Keyder, E.; Hoffmann, J.; and Haslum, P. 2014. Improving delete relaxation heuristics through explicitly represented conjunctions. *Journal of Artificial Intelligence Research* 50:487–533.
- McDermott, D. V. 1999. Using regression-match graphs to control search in planning. *Artificial Intelligence* 109(1-2):111–159.
- Nakhost, H.; Hoffmann, J.; and Müller, M. 2012. Resource-constrained planning: A monte carlo random walk approach. In Bonet, B.; McCluskey, L.; Silva, J. R.; and Williams, B., eds., *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, 181–189. 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.
- Seipp, J., and Helmert, M. 2011. Fluent merging for classical planning problems. In *ICAPS 2011 Workshop on Knowledge Engineering for Planning and Scheduling*, 47–53.
- van den Briel, M.; Benton, J.; Kambhampati, S.; and Vossen, T. 2007. An LP-based heuristic for optimal planning. In Bessiere, C., ed., *Proceedings of the Thirteenth International Conference on Principles and Practice of Constraint Programming (CP'07)*, volume 4741 of *Lecture Notes in Computer Science*, 651–665. Springer-Verlag.