Red-Black Heuristics for Planning Tasks with Conditional Effects

Michael Katz
IBM Research
Yorktown Heights, NY, USA
michael.katz1@ibm.com

Abstract

Red-black planning is a state-of-the-art approach to satisficing classical planning. Red-black planning heuristics are at the heart of the planner *Mercury*, the runner-up of a satisficing track in the International Planning Competition (IPC) 2014 and a major component of four additional planners in IPC 2018, including *Saarplan*, the runner-up in the agile track. *Mercury*'s exceptional performance is amplified by the fact that conditional effects were handled by the planner in a trivial way, simply by compiling them away. Conditional effects, however, are important for classical planning, and many domains require them for efficient modeling.

Consequently, we investigate the possibility of handling conditional effects directly in the red-black planning heuristic function, extending the algorithm for computing red-black plans to the conditional effects setting. We show empirically that red-black planning heuristics that handle conditional effects natively outperform the variants that compile this feature away, improving coverage on tasks where black variables exist by 19%.

Introduction

Delete relaxation heuristics have played a key role in the success of satisficing planning systems (Bonet and Geffner 2001; Hoffmann and Nebel 2001; Richter and Westphal 2010). A well-known pitfall of delete relaxation is its inability to account for repetitive achievements of facts. It has thus been an actively researched question from the outset how to take *some* deletes into account, e. g. (Fox and Long 2001; Gerevini, Saetti, and Serina 2003; Helmert 2004; Helmert and Geffner 2008; Baier and Botea 2009; Cai, Hoffmann, and Helmert 2009; Haslum 2012; Keyder, Hoffmann, and Haslum 2012). The red-black planning framework (Domshlak, Hoffmann, and Katz 2015), where a subset of *red* state variables takes on the relaxed value-accumulating semantics, while the other *black* variables retain the regular semantics, introduced a convenient way of interpolating between fully relaxed and regular planning.

Katz, Hoffmann, and Domshlak (2013b) introduced the red-black framework and conducted a theoretical investigation of tractability of both plan existence and plan generation. In particular, they found that *reversibility* (Chen and Giménez 2010) plays a major role in making red-black plan generation tractable. Following up and exploiting the notion of *invertibility*, Katz, Hoffmann, and Domshlak (2013a) devised practical *red-black plan heuristics*, non-admissible heuristics generated by repairing fully delete-relaxed plans into red-black plans. Observing that this technique often suffers from dramatic over-estimation incurred by following arbitrary decisions taken in delete-relaxed plans, Katz and Hoffmann (2013) refined the approach to rely less on such decisions, yielding a more flexible algorithm that delivers better search guidance. Finally, Katz and Hoffmann (2014b) presented *red-black DAG heuristics* for a tractable fragment characterized by *DAG black causal graphs* and devised some enhancements targeted at making the resulting red-black plans executable in the real task, stopping the search if they succeed in reaching the goal. Red-black DAG heuristics are at the heart of the *Mercury* planner (Katz and Hoffmann 2014a), the runner-up of the sequential satisficing track in the International Planning Competition (IPC 2014). The most recent IPC 2018 had seen four additional participants (not counting variants) that employ red-black planning heuristics, namely *Saarplan* (Fickert et al. 2018), the runner-up of the agile track; *IBaCoP-2018* (Cenamor, de la Rosa, and Fernández 2018), a portfolio-based planner that has *Mercury2014* as one of its components; and *MERWIN* (Katz et al. 2018) and *Cerberus* (Katz 2018), both empowering red-black planning heuristics with the novelty of the heuristic in state (Katz et al. 2017).

All red-black heuristics to this day, however, are defined in a classical planning formalism that does not include conditional effects. Because support of conditional effects is mandatory since IPC 2014, *Mercury* planner did handle conditional effects, but in a naive way. This was done by simply compiling them away in a straightforward fashion, multiplying out the actions (Nebel 2000). Obviously, the number of actions grows exponentially, and thus the straightforward compiling away does not scale well. Nebel (2000) presents an alternative compilation that does not lead to an exponential blow-up in the task size. This compilation, however, does not preserve the delete relaxation. Thus, several delete relaxation-based heuristics were adapted to natively support conditional effects (Haslum 2013; Röger, Pommerening, and Helmert 2014).

Our main goal in this work is to extend the red-black planning framework to tasks with conditional effects. To that end, we extend the red-black planning formalism to support conditional effects. We then generalize the definition of RSE-invertibility (Katz, Hoffmann, and Domshlak 2013a) to tasks with conditional effects and show that the fragment of red-black planning, characterized by *DAG black causal graphs* over RSE-invertible variables, remains tractable. Further, we show how an existing algorithm for solving tasks belonging to this fragment can be adapted to handle conditional effects. Finally, we empirically show the added value of handling the conditional effects directly in the heuristic over compiling them away. We conclude the paper with a discussion of our results and future work.

Background

In order to handle classical planning tasks with conditional effects, we consider the *red-black planning finite-domain representation (RB)* framework (Domshlak, Hoffmann, and Katz 2015), a generalization of both the finite-domain representation and the monotonic finite-domain representation formalisms. We extend the formalism of RB to handle actions with conditional effects. A **red-black (RB)** planning task is a tuple $\Pi = \langle \mathcal{V}^{\mathsf{B}}, \mathcal{V}^{\mathsf{R}}, O, s_0, s_\star \rangle$. $\mathcal{V}^{\mathsf{B}}$ is a set of *black state variables* and $\mathcal{V}^{\mathsf{R}}$ is a set of *red state variables*, where $\mathcal{V}^{\mathsf{B}} \cap \mathcal{V}^{\mathsf{R}} = \emptyset$ and each $v \in \mathcal{V} := \mathcal{V}^{\mathsf{B}} \cup \mathcal{V}^{\mathsf{R}}$ is associated with a finite domain $\mathcal{D}(v)$. The *initial state* s_0 is a complete assignment to $\mathcal{V}$, the *goal* $s_\star$ is a partial assignment to $\mathcal{V}$. Each action o is a pair $\langle \mathit{pre}(o), \mathit{effs}(o) \rangle$, where $\mathit{pre}(o)$ is a partial assignment to $\mathcal{V}$ called *precondition* and $\mathit{effs}(o)$ is a set of *effects*. Each effect $e \in \mathit{effs}(o)$ is a tuple $\langle \mathit{cond}, v, \vartheta \rangle$, where cond is a partial assignment called *effect condition*, $v \in \mathcal{V}$ is the *effect variable*, and $\vartheta \in \mathcal{D}(v)$ is the *effect value*. We often refer to (partial) assignments as sets of *facts*, i. e., variable-value pairs $v = d$. For a partial assignment p, $\mathit{vars}(p)$ denotes the subset of $\mathcal{V}$ instantiated by p. For $\mathcal{V}' \subseteq \mathit{vars}(p)$, $p[\mathcal{V}']$ denotes the value of $\mathcal{V}'$ in p. For the sake of readability, by $\mathit{vars}(\mathit{effs}(o))$ we denote the subset of variables $\mathcal{V}$ that appear in $\mathit{effs}(o)$, that is $\mathit{vars}(\mathit{effs}(o)) = \{v \mid \langle \mathit{cond}, v, \vartheta \rangle \in \mathit{effs}(o)\}$. Also, for the sake of readability, by $\mathit{effs}(o)[\mathcal{V}']$ we refer to the subset of conditional effects that affect variables in $\mathcal{V}'$, that is $\mathit{effs}(o)[\mathcal{V}'] = \{\langle \mathit{cond}, v, \vartheta \rangle \in \mathit{effs}(o) \mid v \in \mathcal{V}'\}$.

A state s assigns each $v \in \mathcal{V}$ a non-empty subset $s[v] \subseteq \mathcal{D}(v)$, where $|s[v]| = 1$ for all $v \in \mathcal{V}^{\mathsf{B}}$. A state s *agrees* with the partial assignment p, denoted by $s \models p$, if $p[v] \in s[v]$ for all $v \in \mathit{vars}(p)$. An action o is applicable in state s if $s \models \mathit{pre}(o)$. An effect $\langle \mathit{cond}, v, \vartheta \rangle \in \mathit{effs}(o)$ *fires* in state s if $s \models \mathit{cond}$. Applying o in s changes the value of v for all firing effects $\langle \mathit{cond}, v, \vartheta \rangle \in \mathit{effs}(o)$ as follows. If $v \in \mathcal{V}^{\mathsf{B}}$, the value is changed to $\{\vartheta\}$. Otherwise (if $v \in \mathcal{V}^{\mathsf{R}}$), the new value of v is $s[v] \cup \{\vartheta\}$. By $s[\![\langle o_1, \dots, o_k \rangle]\!]$ we denote the state obtained from sequential application of $o_1, \dots, o_k$. An action sequence $\langle o_1, \dots, o_k \rangle$ is a *plan* if $s_\star[v] \in s_0[\![\langle o_1, \dots, o_k \rangle]\!][v]$ for all $v \in \mathit{vars}(s_\star)$. Effects $e = \langle \mathit{cond}, v, \vartheta \rangle$ and $e' = \langle \mathit{cond}', v, \vartheta' \rangle$ of the action o are *conflicting* if there exists a state s *reachable* from the initial state such that (a) $s \models \mathit{pre}(o)$ (b) both $s \models \mathit{cond}$ and $s \models \mathit{cond}'$, and (c) $\vartheta \neq \vartheta'$.

Π is a **finite-domain representation (FDR)** planning task if $\mathcal{V}^{\mathsf{R}} = \emptyset$, and is a **monotonic finite-domain representation (MFDR)** planning task if $\mathcal{V}^{\mathsf{B}} = \emptyset$. Plans for MFDR tasks (i. e., for delete-relaxed tasks) can be generated in polynomial time. A key part of many satisficing planning systems is based on exploiting this property for deriving heuristic estimates, via delete-relaxing the task at hand. Generalizing this to red-black planning, the **red-black relaxation** of an FDR task Π relative to $\mathcal{V}^{\mathsf{R}}$ is the RB task $\Pi^{*+}_{\mathcal{V}^{\mathsf{R}}} = \langle \mathcal{V} \setminus \mathcal{V}^{\mathsf{R}}, \mathcal{V}^{\mathsf{R}}, O, s_0, s_\star \rangle$. A plan for $\Pi^{*+}_{\mathcal{V}^{\mathsf{R}}}$ is a **red-black plan** for Π, and the length of a shortest possible red-black plan is denoted $h^{*+}_{\mathcal{V}^{\mathsf{R}}}(\Pi)$. For arbitrary states s, $h^{*+}_{\mathcal{V}^{\mathsf{R}}}(s)$ is defined via the RB task $\langle \mathcal{V} \setminus \mathcal{V}^{\mathsf{R}}, \mathcal{V}^{\mathsf{R}}, O, s, s_\star \rangle$. If $\mathcal{V}^{\mathsf{R}} = \mathcal{V}$, then red-black plans are **relaxed plans**, and $h^{*+}_{\mathcal{V}^{\mathsf{R}}}$ coincides with the optimal delete relaxation heuristic h^+.

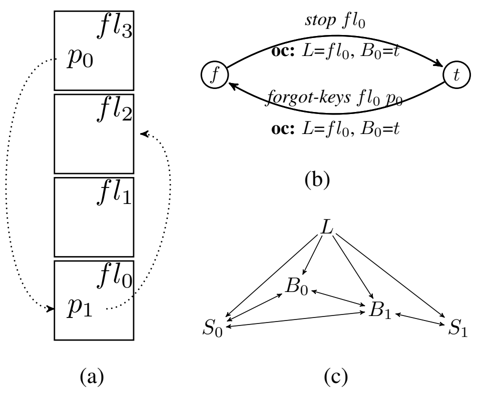

Figure 1: Miconic example (a), a domain transition graph of a variable S_0 (b), and the causal graph (c).

The absence of conflicting effects is typically assumed for FDR tasks. In delete relaxations of these tasks, or more generally in red-black relaxations, such an assumption may not hold anymore, since reachable red-black states may correspond to states not reachable in the original task. It is possible to impose stronger assumptions, ensuring the absence of conflicting effects in red-black relaxations, such as forbidding effects of the same operator that set different values to the same variable. Such assumptions, however, may not hold in practice on existing benchmark sets. To simplify the exposition of our ideas, we resolve this issue by requiring variables with possibly conflicting effects, that is, if there exists an operator with two conditional effects setting that variable to different values, to be marked as red in our red-black relaxations.

We use a slightly modified miconic-simpleadl task with two passengers as our running example. An elevator moves between four floors and needs to move passengers from their original floors to their destination floors. Passenger p_0 is originally at floor fl_3 and needs to get to floor fl_0. Passenger p_1 starts from floor fl_0 and goes to floor fl_2. The example is depicted in Figure 1(a), with floors, passengers and their initial and goal locations shown. There is one variable for elevator location, L, with initial value fl_0, and two variables for each passenger, encoding whether the passenger was boarded, B_i and whether she was served, S_i (with values t and f that stand for true and false). There are actions changing the value of variable L: *up* fl_i fl_j and *down* fl_i fl_j. These actions have one unconditional effect each. In addition, there are actions that stop at a floor, picking up passengers from their origin floors and embarking passengers to their destinations. The *stop* fl_i actions are as follows.

$$\textit{stop } fl_0 = \left\langle \{L{=}fl_0\}, \left\{ \begin{array}{c} \langle \emptyset, B_0, f \rangle, \\ \langle \{B_0{=}t\}, S_0, t \rangle, \\ \langle \{S_1{=}f\}, B_1, t \rangle \end{array} \right\} \right\rangle,$$

$\textit{stop } fl_2 = \langle \{L{=}fl_2\}, \{\langle \emptyset, B_1, f \rangle, \langle \{B_1{=}t\}, S_1, t \rangle\}\rangle$, and

$\textit{stop } fl_3 = \langle \{L{=}fl_3\}, \{\langle \{S_0{=}f\}, B_0, t \rangle\}\rangle$.

Note that there are only actions *stop* fl_0, *stop* fl_2, and *stop* fl_3. There is no action *stop* fl_1, since the floor fl_1 is neither origin nor destination of any passenger.

In addition, there are actions *forgot-keys* fl p with one unconditional effect each, taking the embarked passenger back to the elevator on her destination floor. These actions are as follows.

$$forgot\text{-}keys\ fl_0\ p_0 = \langle \{L{=}fl_0, B_0{=}t, S_0{=}t\}, \{\langle \emptyset, S_0, f \rangle\}\rangle,$$

$$forgot\text{-}keys\ fl_2\ p_1 = \langle \{L{=}fl_2, B_1{=}t, S_1{=}t\}, \{\langle \emptyset, S_1, f \rangle\}\rangle.$$

Focusing on action *stop* fl_0, a straightforward approach of handling conditional effects would multiply out the variables in conditions of the effects. As a result, we obtain four actions:

- $\langle \{L{=}fl_0, B_0{=}f, S_1{=}f\}, \{\langle \emptyset, B_0, f \rangle, \langle \emptyset, B_1, t \rangle\}\rangle$,
- $\langle \{L{=}fl_0, B_0{=}t, S_1{=}f\}, \{\langle \emptyset, B_0, f \rangle, \langle \emptyset, B_1, t \rangle, \langle \emptyset, S_0, t \rangle\}\rangle$,
- $\langle \{L{=}fl_0, B_0{=}f, S_1{=}t\}, \{\langle \emptyset, B_0, f \rangle\}\rangle$, and
- $\langle \{L{=}fl_0, B_0{=}t, S_1{=}t\}, \{\langle \emptyset, B_0, f \rangle, \langle \emptyset, S_0, t \rangle\}\rangle$.

A real optimal plan has length 6 (*up* fl_0 fl_3, *stop* fl_3, *down* fl_3 fl_0, *stop* fl_0, *up* fl_0 fl_2, *stop* fl_2), a relaxed plan has length 5 (no need to go back down to fl_0). If we paint L black, then $h^{*+}_{\mathcal{V}^{\mathsf{R}}}(s_0) = 6$ as desired.

Tractable fragments of red-black planning have been identified using standard structures, **domain transition graph** and **causal graph** (Helmert 2006). The **domain transition graph** $DTG_\Pi(v)$ of a variable $v \in \mathcal{V}$ is a labeled digraph with vertices $\mathcal{D}(v)$. The graph has an arc (d, d') induced by action o if $\langle cond, v, d' \rangle \in \textit{effs}(o)$, and either (i) $pre(o)[v] = d$ or $cond[v] = d$, or (ii) $v \notin vars(pre(o)) \cup vars(cond)$. The arc is labeled with its induced action o and its **outside condition** $pre(o)[\mathcal{V} \setminus \{v\}] \cup cond[\mathcal{V} \setminus \{v\}]$. In contrast to the case of no conditional effects, here we cannot know in advance which effects will fire, and thus the notion of **outside effect** for an edge in the domain transition graph is not well defined. The domain transition graph of the variable S_0 in our example is shown in Figure 1(b). The action labels are above the arcs and the outside conditions (marked by **oc**) are below the arcs.

The **causal graph** CG_Π of Π is a digraph with vertices $\mathcal{V}$. An arc (v, v') is in CG_Π if $v \neq v'$ and there exists an action $o \in O$ such that either (i) the domain transition graph $DTG_\Pi(v')$ of v' has some arc labeled with outside condition on the variable v, or (ii) both $v, v' \in vars(\textit{effs}(o))$.

The **black causal graph** CG^{B}_Π of Π is the sub-graph of CG_Π induced by $\mathcal{V}^{\mathsf{B}}$. Figure 1(c) depicts the causal graph of the example task. If we paint only L black, then the black causal graph is arc-empty. If we paint also S_0 and S_1 black, then the black causal graph is a DAG.

Finally, as mentioned above, the notion of *reversibility* plays a major role in the tractability of red-black planning fragments. A red-black planning task is **reversible** if for every state s reachable from the initial state s_0, there exists a state s' reachable from s such that $s'[v] = s_0[v]$ for all black variables $v \in \mathcal{V}^{\mathsf{B}}$.

Invertibility

The notion of invertibility introduces a sufficient criterion for tractability of red-black planning for tasks with acyclic black causal graphs (Katz, Hoffmann, and Domshlak 2013a; Katz and Hoffmann 2013; Domshlak, Hoffmann, and Katz 2015). This is due to the fact that in the case of acyclic black causal graphs invertibility implies reversibility. The definition of invertibility for tasks without conditional effects is as follows. A DTG arc (d, d') is **relaxed side effects invertible**, **RSE-invertible** for short, if there exists an arc (d', d) with outside condition $\phi' \subseteq \phi \cup \psi$ where ϕ and ψ are the outside condition, respectively, outside effect of (d, d'). A variable v is RSE-invertible if all arcs in $DTG_\Pi(v)$ are RSE-invertible, and an RB task is RSE-invertible if all its black variables are.

The motivation behind the definition is as follows. If the black causal graph is acyclic, every action affects at most one black variable. Since red variables accumulate their values, the only effect we need to invert is the black one. This corresponds to inverting a single arc (d, d') in a domain transition graph. Thus, we need to ensure that for every arc (d, d') traversed in the domain transition graph, there exists a corresponding inverse arc (d', d). That arc must be traversable after (d, d'). Furthermore, since the outside condition ϕ of (d, d') is not changed by traversing the arc, and the outside effect ψ consists of red variables only, and thus can only accumulate new values by traversing the arc, both ϕ and ψ will remain true and can be used as conditions for the inverse arc.

In the presence of conditional effects we need to consider which effects fire, and thus the definition of **RSE-invertible** arcs is adapted as follows.

Definition 1 *Let $v \in \mathcal{V}$ be some variable and o be some action affecting v. Let (d, d') be some arc in the domain transition graph $DTG_\Pi(v)$, induced by $\langle cond, v, d' \rangle \in \textit{effs}(o)$ of action o and let ϕ be its outside condition. The arc (d, d') is **relaxed side effects invertible**, **RSE-invertible** for short, if there exists an induced-by-the-action o' arc (d', d) with outside condition ϕ', such that $\phi' \subseteq \phi \cup \psi$, where $\psi = \{v' = d'' \mid \langle cond', v', d'' \rangle \in \textit{effs}(o), v' \neq v, cond' \subseteq \phi\}$.*

Intuitively, the outside condition of the reverting arc should either hold before traversing the reverted arc or be achieved as a side effect of traversing the arc. For the latter, we can consider only the effects that surely fire.

As before, a variable v is RSE-invertible if all arcs in $\mathit{DTG}_\Pi(v)$ are RSE-invertible, and an RB task is RSE-invertible if all its black variables are. In what follows, we show that the main theorem of Katz, Hoffmann, and Domshlak (2013a), which enabled devising practical red-black heuristics, holds also for tasks with conditional effects.

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

Proof: We follow the proof of Katz, Hoffmann, and Domshlak (2013a) and show that every action application can be undone. Specifically, we show that given a state s and an action o applicable to s, from the state $s[\![\langle o\rangle]\!]$ we can reach a state s' so that $s'[\mathcal{V}^\mathsf{B}] = s[\mathcal{V}^\mathsf{B}]$. If all variables affected by o are red, $s' := s[\![\langle o\rangle]\!]$ fits the requirement. Otherwise, o affects exactly one black variable v. Let (d, d') be the arc in $\mathit{DTG}_\Pi(v)$ that corresponds to the effect $\langle \mathit{cond}, v, d'\rangle \in \mathit{effs}(o)$ fired in s, let (d', d) be the inverse arc, and let o' be the action that induces (d', d) with outside condition ϕ' as in Definition 1. Then o' is applicable in $s[\![\langle o\rangle]\!]$: Using the notations from above, $\mathit{pre}(o') \subseteq \phi' \cup \{(v, d')\}$, where $\phi' \subseteq \phi \cup \psi$. As discussed above, both ϕ and ψ are true in $s[\![\langle o\rangle]\!]$. Clearly, $s' := s[\![\langle o, o'\rangle]\!]$ has the required property. □

In the example, variable L is RSE-invertible, since *up* and *down* actions invert each other. The variables B_0 and B_1 are not RSE-invertible, since edges that correspond to *stop* actions can be inverted only by other *stop* actions. These pairs of *stop* actions correspond to boarding and embarking the passenger, and therefore the elevator location value would not match. The variables S_0 and S_1 are RSE-invertible, as can be seen in Figure 1(b). Note that while the edge from t to f corresponds to a non-conditional effect and thus the outside condition comes entirely from the precondition, the edge from f to t is induced by a conditional effect $\langle \{B_0{=}t\}, S_0, t\rangle$, and thus the outside condition comes partially from the precondition and partially from the effect condition.

Red-Black DAG Heuristics

Katz and Hoffmann (2013) provide an algorithm for RSE-invertible RB tasks with acyclic black causal graphs and no conditional effects. Figure 2 shows the adaptation of Katz and Hoffmann's pseudo-code to conditional effects, with major differences highlighted in red. The original algorithm assumes as input the set R^+ of preconditions and goals on red variables in a fully delete-relaxed plan, i. e., $R^+ = s_\star[\mathcal{V}^\mathsf{R}] \cup \bigcup_{o\in\pi^+} \mathit{pre}(o)[\mathcal{V}^\mathsf{R}]$ where π^+ is a relaxed plan for Π. The adaptation also requires collecting the conditions of the effects that *fire* in the relaxed plan, and therefore R^+ will also include those conditions.

The original algorithm then successively selected achieving actions for R^+, until all these red facts are true. Our algorithm, however, in the presence of conditional effects, instead of selecting an action, is required to select a particular effect of an action. Then, the black condition of that effect should be achieved together with action's black precondition. Throughout the algorithm, R *denotes the set of red facts already achieved by the current red-black plan prefix* π*;* B *denotes the set of black variable values that can be achieved using only red outside conditions from* R*.*

```
Algorithm : RedBlackPlanningCE($\Pi, R^+$)

main
  // $\Pi = \langle \mathcal{V}^\mathsf{B}, \mathcal{V}^\mathsf{R}, O, s_0, s_\star\rangle$
  global $R$,  $B \leftarrow \emptyset$,  $\pi \leftarrow \langle\rangle$
  Update()
  while $R \not\supseteq R^+$
    do
      $O' = \{\langle o, c\rangle \mid o \in O, \langle c, v, \vartheta\rangle \in \mathit{effs}(o),$
                    $\mathit{pre}(o) \cup c \subseteq B \cup R,$
                    $\langle v, \vartheta\rangle \in (R^+ \setminus R)\}$
      Select $\langle o, c\rangle \in O'$
      if $\mathit{pre}(o)[\mathcal{V}^\mathsf{B}] \cup c[\mathcal{V}^\mathsf{B}] \not\subseteq s_0[\![\pi]\!]$
        then $\pi \leftarrow \pi \circ$ Achieve($\mathit{pre}(o)[\mathcal{V}^\mathsf{B}] \cup c[\mathcal{V}^\mathsf{B}]$)
      $\pi \leftarrow \pi \circ \langle o\rangle$
      Update()
  if $s_\star[\mathcal{V}^\mathsf{B}] \not\subseteq s_0[\![\pi]\!]$
    then $\pi \leftarrow \pi \circ$ Achieve($s_\star[\mathcal{V}^\mathsf{B}]$)
  return $\pi$

procedure Update()
  $R \leftarrow s_0[\![\pi]\!][\mathcal{V}^\mathsf{R}]$
  $B \leftarrow B \cup s_0[\![\pi]\!][\mathcal{V}^\mathsf{B}]$
  for $v \in \mathcal{V}^\mathsf{B}$, ordered topologically by the black causal graph
    do $B \leftarrow B \cup \mathit{DTG}_\Pi(v)|_{R\cup B}$

procedure Achieve($g$)
  $s_0^\mathsf{B} \leftarrow s_0[\![\pi]\!][\mathcal{V}^\mathsf{B}]$
  $s_\star^\mathsf{B} \leftarrow g$
  $O^\mathsf{B} \leftarrow \{o^\mathsf{B} \mid o \in O, \mathit{pre}(o) \subseteq R \cup B, \bigcup_{\langle c,v,\vartheta\rangle\in \mathit{effs}(o)[\mathcal{V}^\mathsf{B}]} c \subseteq R \cup B,$
          $\mathit{pre}(o^\mathsf{B}) = \mathit{pre}(o)[\mathcal{V}^\mathsf{B}],$
          $\mathit{effs}(o^\mathsf{B}) = \{\langle c[\mathcal{V}^\mathsf{B}], v, \vartheta\rangle \mid \langle c, v, \vartheta\rangle \in \mathit{effs}(o)[\mathcal{V}^\mathsf{B}]\}\}$

  $\langle o_1'^\mathsf{B}, \ldots, o_k'^\mathsf{B}\rangle \leftarrow$ an FDR plan for $\Pi^\mathsf{B} = \langle \mathcal{V}^\mathsf{B}, O^\mathsf{B}, s_0^\mathsf{B}, s_\star^\mathsf{B}\rangle$

  return $\langle o_1', \ldots, o_k'\rangle$
```

Figure 2: Red-black planning algorithm.

How the algorithm works: For each selected achieving action and effect pair (or for the goal), if the black variable values do not match the precondition of the selected action and effect condition (or the goal), the algorithm attempts to achieve the aforementioned partial state. For that, Achieve(g) is called, which finds a sequence of actions achieving the partial state by solving the black subtask Π^B with invertible variables.

Katz and Hoffmann (2014b) present a simple algorithm that solves the black subtask: Starting at the leaf variables and working up to the roots, the plan is constructed by augmenting the partial plan with plan fragments that correspond to paths in domain transition graphs, bringing the supporting variables into place [1]. They show the algorithm runtime to be polynomial in the size of Π^B and the length of the plan

[1]A similar algorithm was mentioned, but not used, by Helmert (2006)

returned, which, although worst-case exponential in the size of Π^{B}, is practically efficient. As conditional effects correspond to edges in domain transition graphs, the algorithm works verbatim for tasks with conditional effects.

Although we impose a restriction to avoid conflicting effects on black variables, it is worth mentioning that our adapted algorithm can handle such effects, in the sense that it will produce a sequence of actions that corresponds to a red-black plan if the "right" effect is chosen to resolve the conflict. In ACHIEVE(g), recall that the effects are restricted to a single variable, and thus actions with conflicting effects can simply be split into multiple actions. In UPDATE(), the update is performed using DTG edges, and thus is not affected by conflicting effects. Finally, in the main procedure, an action and its effect are selected, and therefore the main procedure is also not affected by conflicting effects.

Experimental Evaluation

In order to evaluate the benefit of natively supporting conditional effects in red-black planning heuristics, we adapted the existing implementation of red-black planning heuristics on top of the current Fast Downward framework (Helmert 2006). We compared our native support (NS) to the original implementation on top of a transformation that multiplies out conditional effects (MO), a baseline for our comparison. We perform a greedy best first search with a single queue, ordered by the red-black heuristic. The heuristic is obtained by solving a red-black planning task with a black DAG causal graph. The red-black planning task is obtained by coloring the RSE-invertible variables black as long as the black part is a DAG (Domshlak, Hoffmann, and Katz 2015).

The experiments were performed on Intel(R) Xeon(R) CPU E7-8837 @2.67GHz machines, with the time and memory limit of 30min and 2GB, respectively. We performed our evaluation on the existing set of benchmark domains with conditional effects. Only a handful of domains from previous International Planning Competitions (IPC) have conditional effects but no axioms. Therefore, following Haslum (2013), we also use problems generated by the conformant-to-classical planning compilation (T0) (Palacios and Geffner 2009) and the finite-state controller synthesis compilation (FSC) (Bonet, Palacios, and Geffner 2009). In addition, following Röger, Pommerening, and Helmert (2014), we use the briefcase world domain from the IPP benchmark collection (Köhler 1999) and the Miconic simpleadl version from the benchmark set of the International Planning Competition (IPC2000), as it has conditional effects but no derived predicates after grounding with Fast Downward. Finally, we use the domains from the most recent IPC 2018. This results in total of 679 tasks in 32 domains, shown in Table 1. The table also depicts the number of tasks per domain, as well as the per-domain summary of the number of tasks for which a relevant SAS$^+$ representation could be constructed and the number of tasks where at least one RSE-invertible variable was found. There are 28 cases, all in IPC2018 domains where the translator was not able to finish translating PDDL to SAS$^+$ representation, namely four tasks in caldera-sat18, six in flashfill-sat18, 17 in organic-synthesis-sat18, and one in organic-synthesis-split-sat18, for both our approach and the baseline. In all other cases, when the tasks could not be constructed, it was due to the transformation that multiplied out conditional effects resulting in a task that was too large to fit into memory. This happened in 243 out of the total 679 tasks. Note that there is no way around creating the transformed task in order to check RSE-invertibility of variables in the transformed task. Then, an additional check is performed, which invertible variables may have conflicting effects in the red-black relaxation, to ensure these variables are marked as red. This happens in two tasks of the t0-adder domain. If no non-conflicting RSE-invertible variables are found, red-black heuristics are effectively equivalent to the FF heuristic (Hoffmann and Nebel 2001). Importantly, in such cases, for the baseline approach, the FF heuristic would be constructed

		multiply out		native support	
Domain	#	Constr.	Inv.	Constr.	Inv.
briefcaseworld	50	10	10	50	50
cavediving-14-adl	20	20	20	20	20
citycar-sat14-adl	20	20	0	20	0
fsc-blocks	14	0	0	14	0
fsc-grid-a1	16	1	0	16	0
fsc-grid-a2	2	1	0	2	0
fsc-grid-r	16	0	0	16	0
fsc-hall	2	1	0	2	0
fsc-visualmarker	7	0	0	7	0
gedp-ds2ndp	24	4	4	24	0
miconic-simple	150	150	150	150	150
t0-adder	2	0	0	2	2
t0-coins	30	20	15	30	30
t0-comm	25	25	0	25	0
t0-grid-dispose	15	0	0	15	15
t0-grid-push	5	0	0	5	5
t0-grid-trash	1	0	0	1	1
t0-sortnet	5	0	0	5	0
t0-sortnet-alt	6	1	0	6	0
t0-uts	29	13	0	29	4
agricola18	20	20	0	20	0
caldera18	20	16	0	16	0
caldera-sp18	20	20	0	20	0
data-network18	20	20	20	20	20
flashfill18	20	1	0	14	0
nurikabe18	20	11	11	20	20
organic-synt18	20	3	0	3	0
organic-synt-sp18	20	19	2	19	2
settlers18	20	0	0	20	0
snake18	20	20	0	20	0
spider18	20	20	0	20	0
termes18	20	20	20	20	20
Sum	679	436	252	651	339

Table 1: Per-domain summary of the number of tasks (#); number of constructed (Constr.) and number of tasks with invertible variables (Inv.) for both approaches.

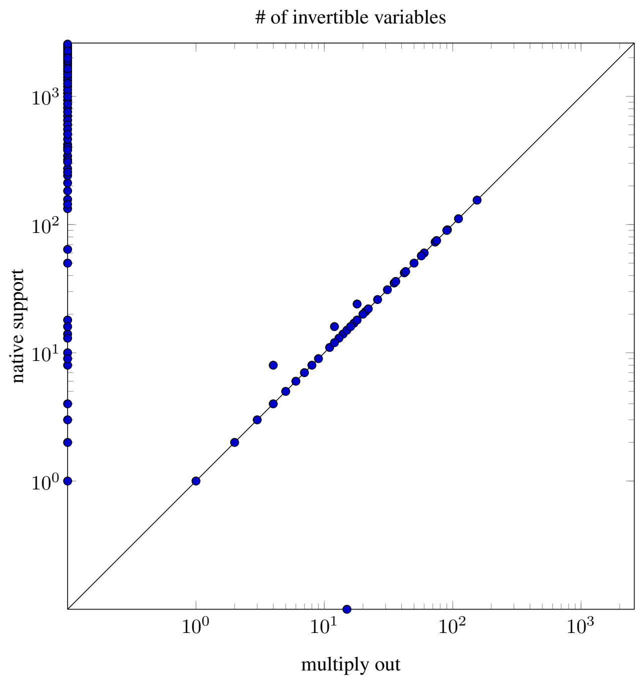

Figure 3: Comparison of the number of RSE-invertible variables with the baseline multiplying out conditional effects and the native support for conditional effects.

Domain	#	Inv.	mult. out	native supp.
briefcaseworld	50	50	10	**50**
cavediving-14-adl	20	20	**7**	**7**
gedp-ds2ndp	24	4	0	**4**
miconic-simple	150	150	**150**	**150**
t0-coins	30	30	**20**	**20**
t0-grid-dispose	15	15	0	**15**
t0-grid-push	5	5	0	**3**
t0-grid-trash	1	1	0	0
t0-uts	29	4	**3**	**3**
data-network18	20	20	0	0
nurikabe18	20	20	5	**6**
organic-synt-sp18	20	2	0	0
termes18	20	20	**13**	**13**
Sum	404	341	208	**271**

Table 2: Per-domain coverage for domains with invertible variables, tasks for which at least one approach found invertible variables. Best performers are marked in bold.

on top of the transformed task. Looking at the table, there are 343 tasks in 14 domains where either of the two approaches found any RSE-invertible variables. In what follows, we restrict our attention to these tasks.

Figure 3 compares the number of RSE-invertible variables for each of these tasks. Each point corresponds to a task, showing the number of RSE-invertible variables for the multiply-out approach and for the native one. Observe that the number of RSE-invertible variables mostly increases when moving from compiling away conditional effects to natively supporting them. The points on the "multiply out" axis correspond to four tasks in gedp-ds2ndp domain, where the multiply-out transformation resulted in a task with fifteen RSE-invertible variables, while the original task has no RSE-invertible variables under our new definition of RSE-invertibility in the presence of conditional effects. For the reverse case ("native support" axis), there are 91 tasks where there could not be found any RSE-invertible variables in the transformed task,[2] and there was at least one RSE-invertible variable under our new definition, allowing us to use the red-black planning heuristic, rather than the base FF heuristic.

Turning our attention to the heuristics performance,[3] Table 2 shows the per-domain coverage, comparing our suggested approach to the baseline. The second column describes the overall number of instances in each domain, while the third column shows the number of instances on which at least one of the approaches found invertible variables (Inv.). The last two colums report the number of solved tasks for the compared approaches. The table clearly shows the benefit of handling conditional effects natively in the heuristic – the coverage never decreases, remains the same in eight out of 12 domains, and increases by 63 instances in five domains, namely by 40, 15, 4, 3, and 1 in briefcaseworld, t0-grid-dispose, gedp-ds2ndp, t0-grid-push, and nurikabe-sat18, respectively. Note that if we compare the coverage on all instances of all 32 domains, without considering whether invertible variables exist or not, the baseline solves 314 out of 679 tasks, while our approach solves 438. However, on tasks where no RSE-invertible variables were found, such comparison would essentially be between the FF heuristics with and without task transformation. While such comparison is interesting by itself, it is outside the scope of our current work.

In order to show a per-instance comparison, Figure 4 compares the number of heuristic evaluations performed until a solution is found. First, observe that out of the 63 points on the right border that correspond to tasks in our restricted set not solved with the baseline approach, but solved with our approach, there are 57 that are now solved after up to 10000 evaluations. An additional six tasks are going as high as 36911 evaluations. There are only six tasks, all in briefcaseworld, where the performance got negligably worse (by at most 77 evaluated nodes in the extreme case), remained the same on 58 tasks, and got better on 208 tasks, including the 63 tasks where the baseline was not able to solve the task at all. The other 145 tasks, where both solved the task and the native support performed better, are from the miconic-simpleadl domain (125 tasks), t0-coins (17 tasks), and t0-uts (three tasks), with the improvement in t0-coins being negligable. This brings us to the remaining 128 tasks. Arguably, the most important feature of red-black planning

[2]In some of these cases, the transformed task could not even be created, due to hitting the memory bound.

[3]We exclude from our task set the two tasks of t0-adder where all invertible variables appear in conditional effects that may conflict in the red-black relaxation and thus are marked as red.

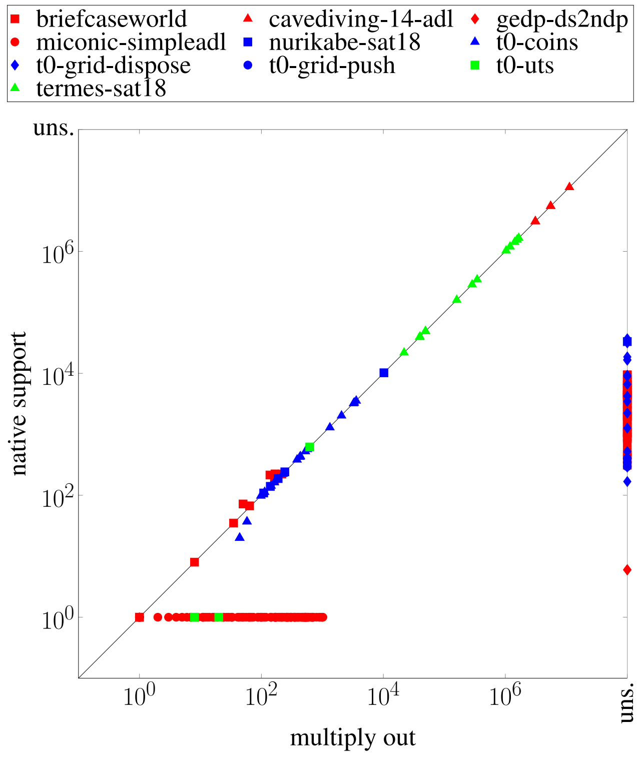

Figure 4: Domain-wise comparison of the number of evaluations performed with the baseline multiplying out conditional effects and the native support for conditional effects.

heuristics is the similarity of red-black plans to real plans. In many cases, a red-black plan for the initial state is already a real plan and thus no search needs to be performed at all. With the baseline, this happens in 27 tasks, 25 tasks in miconic-simpleadl and two tasks in briefcaseworld. With our approach of supporting conditional effects natively, this happens in all the aforementioned cases, and more. In total, the real plan is found in the initial state for 155 tasks. The additional 128 tasks are partitioned to 125 tasks of miconic-simpleadl and three tasks in t0-uts. Consequently, all the 150 tasks of the miconic-simpleadl domain are now solved without search. These 155 tasks, that are solved by our approach without performing any search, correspond to the points on the horizontal line in Figure 4, for the number of evaluations equal to one.

Finally, to measure the heuristic computation time difference, Figure 5 shows the per-instance total time comparison. Here as well, the picture is clear. The vast majority of points are under the diagonal, with the points on the rightmost border correspond to the tasks solved by our method but not by the multiply out approach. Many of the tasks that are not solved by the baseline approach are now solved in under one second. Focusing now on tasks that are solved by both approaches, while in most cases the improvement is moderate, there are tasks where the total time is improved by up to three orders of magnitude. This clearly demonstrates the practical benefit of supporting conditional effects natively in

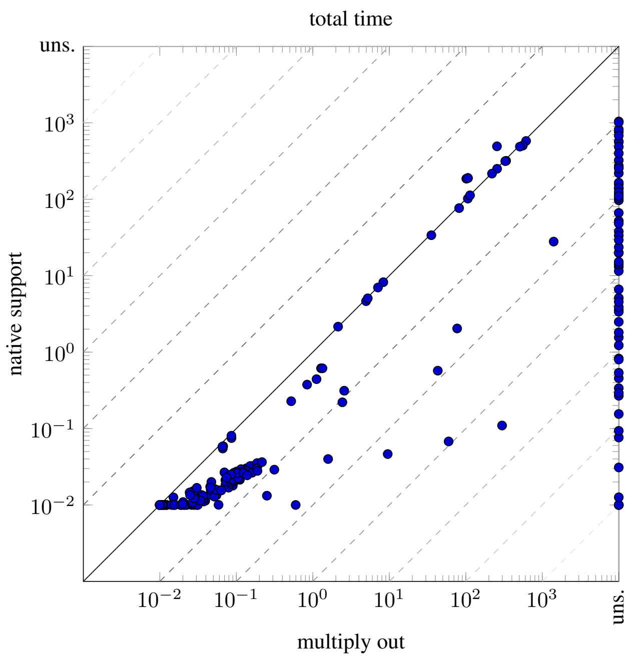

Figure 5: Comparison of the total time until a solution is found with the baseline multiplying out conditional effects and the native support for conditional effects.

red-black planning heuristics over the previous methods of compiling conditional effects away.

Conclusions and Future Work

We have shown how to adapt red-black planning to support conditional effects, an important modeling feature in planning. To that end, we have adapted the formalism of red-black planning to include conditional effects and have shown how to derive practical red-black planning heuristics. For that, we extended the definition of invertibility of variables and adapted the existing algorithms for tractable red-black planning for the classical formalism to work in the presence of conditional effects. To measure the benefit of natively supporting conditional effects in the red-black planning heuristic, we performed an extensive experimental evaluation. We compared the native support for conditional effects in red-black planning heuristics with the existing naive way of supporting conditional effects by compiling them away. Our evaluation clearly shows the benefit of supporting conditional effects natively.

For future work, we intend to explore natively supporting additional non-classical features in red-black planning, such as axioms and derived predicates (McDermott et al. 1998; Thiébaux, Hoffmann, and Nebel 2005). For conditional effects, one issue not fully covered in the current work is conflicting effects. We intend to further extend the framework and adapt our proofs to cover conflicting effects, allowing us to paint their respective variables black. Another important direction is to extend the fully unrelaxed applicability of the resulting red-black plans. Current red-black planning heuristics face two choice points. The first: what action and effect to select to achieve some currently unachieved but required red fact; the second: how to achieve the precondition and the condition of the action effect selected. Preferring real paths in the second choice point has significantly improved unrelaxed applicability. Not much was done, however, for the first choice point, except for preferring action effects that can fire. Selecting these action effects in a way that can improve unrelaxed applicability is a very promising research direction with high potential for practical value.

References

Baier, J. A., and Botea, A. 2009. Improving planning performance using low-conflict relaxed plans. In *Proc. ICAPS 2009*, 10–17.

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

Bonet, B.; Palacios, H.; and Geffner, H. 2009. Automatic derivation of memoryless policies and finite-state controllers using classical planners. In *Proc. ICAPS 2009*, 34–41.

Cai, D.; Hoffmann, J.; and Helmert, M. 2009. Enhancing the context-enhanced additive heuristic with precedence constraints. In *Proc. ICAPS 2009*, 50–57.

Cenamor, I.; de la Rosa, T.; and Fernández, F. 2018. IBaCoP-2018 and IBaCoP2-2018. In *IPC-9 planner abstracts*, 9–10.

Chen, H., and Giménez, O. 2010. Causal graphs and structurally restricted planning. *Journal of Computer and System Sciences* 76(7):579–592.

Domshlak, C.; Hoffmann, J.; and Katz, M. 2015. Red-black planning: A new systematic approach to partial delete relaxation. *AIJ* 221:73–114.

Fickert, M.; Gnad, D.; Speicher, P.; and Hoffmann, J. 2018. Saarplan: Combining saarland's greatest planning techniques. In *IPC-9 planner abstracts*, 11–16.

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

Gerevini, A. E.; Saetti, A.; and Serina, I. 2003. Planning through stochastic local search and temporal action graphs in LPG. *JAIR* 20:239–290.

Haslum, P. 2012. Incremental lower bounds for additive cost planning problems. In *Proc. ICAPS 2012*, 74–82.

Haslum, P. 2013. Optimal delete-relaxed (and semi-relaxed) planning with conditional effects. In *Proc. IJCAI 2013*, 2291–2297.

Helmert, M., and Geffner, H. 2008. Unifying the causal graph and additive heuristics. In *Proc. ICAPS 2008*, 140–147.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *Proc. ICAPS 2004*, 161–170.

Helmert, M. 2006. The Fast Downward planning system. *JAIR* 26:191–246.

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

Katz, M., and Hoffmann, J. 2013. Red-black relaxed plan heuristics reloaded. In *Proc. SoCS 2013*, 105–113.

Katz, M., and Hoffmann, J. 2014a. Mercury planner: Pushing the limits of partial delete relaxation. In *IPC-8 planner abstracts*, 43–47.

Katz, M., and Hoffmann, J. 2014b. Pushing the limits of partial delete relaxation: Red-black DAG heuristics. In *ICAPS 2014 Workshop on Heuristics and Search for Domain-independent Planning*, 40–44.

Katz, M.; Lipovetzky, N.; Moshkovich, D.; and Tuisov, A. 2017. Adapting novelty to classical planning as heuristic search. In *Proc. ICAPS 2017*, 172–180.

Katz, M.; Lipovetzky, N.; Moshkovich, D.; and Tuisov, A. 2018. Merwin planner: Mercury enchanced with novelty heuristic. In *IPC-9 planner abstracts*, 53–56.

Katz, M.; Hoffmann, J.; and Domshlak, C. 2013a. Red-black relaxed plan heuristics. In *Proc. AAAI 2013*, 489–495.

Katz, M.; Hoffmann, J.; and Domshlak, C. 2013b. Who said we need to relax *all* variables? In *Proc. ICAPS 2013*, 126–134.

Katz, M. 2018. Cerberus: Red-black heuristic for planning tasks with conditional effects meets novelty heuristic and enchanced mutex detection. In *IPC-9 planner abstracts*, 47–51.

Keyder, E.; Hoffmann, J.; and Haslum, P. 2012. Semi-relaxed plan heuristics. In *Proc. ICAPS 2012*, 128–136.

Köhler, J. 1999. Handling of conditional effects and negative goals in IPP. Technical Report 128, University of Freiburg, Department of Computer Science.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL – The Planning Domain Definition Language – Version 1.2. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, Yale University.

Nebel, B. 2000. On the compilability and expressive power of propositional planning formalisms. *JAIR* 12:271–315.

Palacios, H., and Geffner, H. 2009. Compiling uncertainty away in conformant planning problems with bounded width. *JAIR* 35:623–675.

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

Röger, G.; Pommerening, F.; and Helmert, M. 2014. Optimal planning in the presence of conditional effects: Extending LM-Cut with context splitting. In *Proc. ECAI 2014*, 765–770.

Thiébaux, S.; Hoffmann, J.; and Nebel, B. 2005. In defense of PDDL axioms. *AIJ* 168(1–2):38–69.