What Does it Take to Render $h^+(\Pi^C)$ Perfect?

Jörg Hoffmann and **Marcel Steinmetz**
Saarland University
Saarbrücken, Germany
hoffmann@cs.uni-saarland.de, s9mrstei@stud.uni-saarland.de

Patrik Haslum
The Australian National University & NICTA
Canberra, Australia
patrik.haslum@anu.edu.au

Abstract

It is well-known that $h^+(\Pi^C)$ is perfect in the limit, i. e., we can always choose C so that $h^+(\Pi^C) = h^*$. But the proof is trivial (select C as the set of all conjunctions), and completely ignores the actual power of $h^+(\Pi^C)$, basically pretending that h^+ is the same as h^1. It is thus interesting to ask: *Can we characterize the power of $h^+(\Pi^C)$ more accurately? How large does C have to be, under which circumstances?*

We present first results towards answering these questions. We introduce a "direct" characterization of $h^+(\Pi^C)$, in terms of equations, not employing a compilation step. We identify a first tractable fragment (similar to fork causal graphs) where size-2 conjunctions suffice to render $h^+(\Pi^C)$ perfect. We present results comparing $h^+(\Pi^C)$ to alternative partial delete relaxation methods (red-black planning and fluent merging). We finally present a number of wild speculations as to what might be interesting to investigate in the future.

Disclaimer: We are enthusiastic about the research direction, but our work as yet raises far more questions than answers. We think that HSDIP is a great forum to discuss this big riddle, and we hope that other researchers may feel compelled to look at it.

Introduction

Haslum's (2009) work on compiling fact conjunctions into the planning task, allowing to simulate h^m via h^1, led a few years later to a partial delete relaxation method able to interpolate all the way between h^+ and h^*: The Π^C compilation (Haslum 2012) allows to select any subset C of fact conjunctions, and outputs a compiled task Π^C so that $h^+(\Pi^C)$ is admissible and perfect in the limit, i. e., we can always choose C so that $h^+(\Pi^C) = h^*$.

The size of Π^C is worst-case exponential in $|C|$, which has been solved via a slightly weaker compilation Π^C_{ce} (Keyder, Hoffmann, and Haslum 2012) exploiting conditional effects, but for the sake of simplicity we abstract from that issue here and consider only $h^+(\Pi^C)$. Our primary objective is to *scratch the itch that results from reading the proof of $h^+(\Pi^C)$ convergence*: The proof is derived from the inequalities (a) $h^m \leq h^1(\Pi^C)$ when C contains all m-tuples, and (b) $h^1(\Pi^C) \leq h^+(\Pi^C)$. In other words, $h^+(\Pi^C)$ convergence is inherited from that of h^m which is completely impractical (set m to the total number of facts). The proof completely ignores the actual added power of $h^+(\Pi^C)$, namely (a) being able to choose C freely, as well as (b) *the advantage of h^+ over h^1!*

Another way to say this is that our theory so far is completely disconnected from practice, where of course $h^+(\Pi^C)$ with C being all fact pairs will in most cases be a *much* better heuristic than h^2. Can we reconcile the theory with practice?[1] Can we characterize more accurately the circumstances under which $h^+(\Pi^C)$ becomes perfect? When does that require C to be exponentially large, and when is polynomial-size C enough? Can we exploit such insights to choose C in a targeted manner?

We believe that these are interesting research questions. We are not so sure about the significance of our answers so far. Certainly, we are nowhere near answering the last question, i. e., it is unclear how (and whether at all) our results so far can be made useful in practice. Our hope is that other HSDIP researchers will find our questions and partial answers inspiring, leading to interesting discussions and, eventually, better progress on this subject.

After preliminaries (notations, Π^C compilation), we make a few simple observations about the size of C depending on the value of h^*. We then introduce a "direct" characterization of $h^+(\Pi^C)$, in terms of equations, that does not need to go via a compilation step (as a side effect, this also yields a somewhat novel view on h^+). Towards an analysis of "tractable fragments", i. e., planning sub-classes in which polynomial-size C suffices to render $h^+(\Pi^C)$ perfect, we introduce a first such fragment similar to fork causal graphs, where size-2 conjunctions suffice. We present results comparing $h^+(\Pi^C)$ to alternative partial delete relaxation methods, namely red-black planning (Katz, Hoffmann, and Domshlak 2013b; 2013a; Katz and Hoffmann 2013) and fluent merging (van den Briel, Kambhampati, and Vossen 2007; Seipp and Helmert 2011). We close the paper ("conclusion") with a number of wild speculations as to what might be interesting to investigate in the future.

[1] To be fair, it should be said that, while the above proof is given by both Haslum (2012) and Keyder et al. (2012), Haslum also gives an alternative proof via convergence of iterative relaxed plan refinement. The latter proof, however, involves excluding all flawed relaxed plans one-by-one, which does not seem to be any more directly illuminating regarding the practical power of $h^+(\Pi^C)$.

Preliminaries

In difference to prior works on Π^C, we use an FDR framework. Planning tasks are tuples $\Pi = (V, A, I, G)$ of variables, actions, initial state, and goal, each action a being a pair $(pre(a), \mathit{eff}(a))$ as usual. We consider uniform costs only (i. e., all action costs are 1). We refer to variable/value pairs as facts, and we perceive (partial) variable assignments as sets of facts. The set of all facts in a planning task is denoted F. We will from now on assume this setup tacitly, i. e., we won't repeat it in formal claims etc.

We say that a set X of facts is *consistent* if there does not exist a variable v so that X contains more than one value for v. Otherwise, we say that X is *contradictory*.

When we talk about heuristic functions h, we mean their value $h(I)$ in the initial state (i. e., for the moment we do not consider rendering$h^+(\Pi^C)$ perfect across all states). By $h(\Pi')$, we denote a heuristic function for Π whose value is given by applying h in a modified task Π'. It is sometimes of advantage to make explicit that h is a heuristic computed on Π itself; we will denote that by $h(\Pi)$.

The delete relaxation in FDR, and thus h^+ in our setup, is defined by interpreting states as fact sets allowed to be contradictory, and where applying action a to state s yields the outcome state $s \cup \mathit{eff}(a)$; the initial state is the same as before. Intuitively, this just means that we are interpreting the effect list $\mathit{eff}(a)$ exactly like the add lists in STRIPS.

The Π^C compilation and its relatives are based on introducing π-*fluents* of the form π_c, each of which represents a conjunction c of facts. In the context of FDR, π_c is a Boolean variable; we will treat it like a STRIPS fact, e. g., we write $\pi_c \in s$ if π_c is true in s, and $\pi_c \notin s$ otherwise. We identify conjunctions with fact sets. For fact sets X, we use the shorthand $X^C = X \cup \{\pi_c \mid c \in C \wedge c \subseteq X\}$. In other words, X^C consists of the set of facts X itself, together with all facts π_c representing conjunctions $c \in C$ such that $c \subseteq X$. With this, Π^C can be defined as follows:

Definition 1 (The Π^C compilation) *Given a set C of conjunctions, Π^C is the planning task (V^C, A^C, I^C, G^C), where $V^C = V \cup \{\pi_c \mid c \in C\}$, and A^C contains an action $a^{C'}$ for every pair $a \in A$, $C' \subseteq C$ such that*

- *for all $c' \in C'$, $\mathit{eff}(a) \cap c' \neq \emptyset$, and $\mathit{eff}(a) \cup c'$ is consistent.*

Here, $a^{C'}$ is given by

- $pre(a^{C'}) = (pre(a) \cup \bigcup_{c' \in C'} (c' \setminus \mathit{eff}(a)))^C$, *and*
- $\mathit{eff}(a^{C'}) = \mathit{eff}(a) \cup \{\pi_{c'} \mid c' \in C'\}$.

This definition, apart from using FDR instead of STRIPS, diverges from Haslum's (2012) in three ways. We do not demand C' to be "downward closed", i. e., to contain all c' subsumed by C'; we do not automatically include $\pi_{c'}$ facts relying on non-deleted preconditions; and we do not include any delete effects. None of these changes have any consequences for the results we present. The first just introduces some superfluous actions, the second change means that we need to include these $\pi_{c'}$ facts explicitly into C', and the third change is made as such effects are irrelevant to h^+ which is our exclusive focus here.

We denote by $C^m := \{c \subseteq F \mid |c| \leq m\}$ the set of all size-$\leq m$ conjunctions. We denote Π^C with $C = C^m$ by Π^{Cm}. We will often consider Π^{Cm} only, abstracting from the ability of Π^C to choose an arbitary C. The underlying intuition/hypothesis is that, in most cases, this abstraction level will suffice to determine the desired distinction between polynomial-size C and exponentially large C.

We will sometimes employ regression-based characterizations of h^* and h^+. The *regression of fact set g over action a*, $R(g, a)$, is defined if $\mathit{eff}(a) \cap g \neq \emptyset$ and $\mathit{eff}(a) \cup g$ is consistent.[2] If $R(g, a)$ is defined, then $R(g, a) = (g \setminus \mathit{eff}(a)) \cup pre(a)$; otherwise, we write $R(g, a) = \bot$.

Obviously, $h^* = h^*(G)$ where $h^*(g)$, for a set g of facts, is the function that satisfies $h^*(g) =$

$$\begin{cases} 0 & g \subseteq I \\ 1 + \min_{a \in A, R(g,a) \neq \bot} h^*(R(g, a)) & \text{otherwise} \end{cases} \quad (1)$$

Similarly, $h^+ = h^+(G)$ where $h^+(g)$, for a set g of facts, is the function that satisfies $h^+(g) =$

$$\begin{cases} 0 & g \subseteq I \\ 1 + \min_{a \in A, \mathit{eff}(a) \cap g \neq \emptyset} & \\ \quad h^+((g \setminus \mathit{eff}(a)) \cup pre(a)) & \text{otherwise} \end{cases} \quad (2)$$

Under the delete relaxation, a sub-goal g can be achieved through action a iff part of it is achieved by a's effect, regardless of any contradictions that may be present.

Remember finally that h^m is defined as $h^m = h(G)$ where $h^m(g)$, for a set g of facts, is the function that satisfies $h^m(g) =$

$$\begin{cases} 0 & g \subseteq I \\ 1 + \min_{a \in A, R(g,a) \neq \bot} h^m(R(g, a)) & |g| \leq m \\ \max_{g' \subseteq g, |g'| \leq m} h^m(g') & \text{otherwise} \end{cases} \quad (3)$$

The Size of C vs. the Value of h^*

A possible starting point for thinking about the size of C is comparing it to the value of h^*. A trivial observation is immediately made:

Proposition 1 *If $h^+(\Pi^C) < \infty$, then $h^+(\Pi^C) - h^+(\Pi) \leq |C|$.*

This holds simply because a relaxed plan needs to achieve every fact (including π-fluents) at most once. We get:

Proposition 2 *If, in a planning task family $\{\Pi_n\}$ whose size relates polynomially to n, h^* grows exponentially in n, then so must C in order to render $h^+(\Pi^C)$ perfect.*

Denoting by F_n the set of facts in Π_n, with Proposition 1 we have $|C| >= h^*(\Pi_n) - F_n$, showing this claim.

Proposition 2 opens the question whether there exist cases with polynomial h^*, but where super-polynomial growth of C is needed nevertheless. The answer is a qualified "yes":

[2] It is sometimes required also that $(g \setminus \mathit{eff}(a)) \cup pre(a)$ is consistent. The two definitions are equivalent as contradictory subgoals will be unreachable anyhow. We use the simpler definition as it is closer to h^+ and its relatives.

Proposition 3 *There exist planning task families $\{\Pi_n\}$ whose size relates polynomially to n, where h^* grows polynomially in n for solvable tasks, but where (unless **P=NP**) C must grow super-polynomially in n in order to render $h^+(\Pi^C)$ perfect.*

Proof: Simply encode SAT into a planning task whose size relates polynomially to the number n of clauses, and where a plan consists of choosing a value for each variable, then evaluating that all clauses are satisfied. Then h^* grows polynomially in n for solvable tasks. Assume that polynomial-sized C suffices to render $h^+(\Pi^C)$ perfect. As relaxed plan existence is equivalent to $h^1 < \infty$, we could then in polynomial time decide whether or not $h^* = h^+(\Pi^C) < \infty$, yielding **P=NP**. ∎

Proposition 3 is only a "qualified" yes because its setup is not fair: Whereas we require h^* to grow polynomially only on solvable tasks (ignoring the ∞ cases), we require $h^+(\Pi^C)$ to be perfect *everywhere*, including the ∞ cases. For solvable SAT instances, $h^+(\Pi^C)$ might very well get perfect with small C already – or, at least, the current proof makes no statement about that.

Open Question 1 *Do there exist families of solvable tasks $\{\Pi_n\}$ whose size relates polynomially to n, where h^* grows polynomially in n, but where C must grow super-polynomially in n to render $h^+(\Pi^C)$ perfect? Most extremely, where on top of this h^* can be computed in polynomial time?*

We conjecture that the answer to this one is "yes", but our proof attempts so far did not succeed. Note here that the simple proof of Proposition 3 above relies crucially on needing to test only whether $h^+(\Pi^C) = \infty$, which can be done in polynomial time. On solvable tasks, as demanded in Open Question 1, perfect $h^+(\Pi^C)$ will be finite, so even for small C it is **NP**-hard to decide whether a given bound is met. For illustration: Say that, in the proof of Proposition 3, we introduce a "side route" in the SAT encoding, rendering unsat cases solvable but via a longer plan. Then we can still read off sat vs. unsat from perfect $h^+(\Pi^C)$, but we can no longer do so in polynomial time, so do not get a contradiction to the hardness of SAT.

Characterizing $h^+(\Pi^C)$ w/o Compilation

Trying to lead proofs about $h^+(\Pi^C)$, it can be annoying that one always has to do the mapping from original task to compiled task first. To make do without this, we now characterize $h^+(\Pi^C)$ directly in terms of the original planning task. Focusing on Π^{Cm} only for the moment, $h^+(\Pi^{Cm})$ can be understood as the following hybrid of h^m and h^+:

Definition 2 (Marrying h^m with h^+: h^{m+}) *The* critical path delete relaxation heuristic *h^{m+} is defined as $h^{m+} := h^{m+}(\{G\})$, where $h^{m+}(\mathcal{G})$, for a set $\mathcal{G}$ of fact sets, is the function that satisfies $h^{m+}(\mathcal{G}) =$*

$$\begin{cases} 0 & \forall g \in \mathcal{G} : g \subseteq I \\ 1 + \min_{a \in A, \emptyset \neq \mathcal{G}' \subseteq \{g \in \mathcal{G} \mid R(g,a) \neq \bot\}} & \\ \quad h^{m+}((\mathcal{G} \setminus \mathcal{G}') \cup \{\bigcup_{g \in \mathcal{G}'} R(g,a)\}) & \forall g \in \mathcal{G} : |g| \leq m \\ h^{m+}(\bigcup_{g \in \mathcal{G}} \{g' \subseteq g \mid |g'| \leq m\}) & \text{otherwise} \end{cases}$$

The underlying idea here is that the delete relaxation can be understood as allowing to achieve sub-goals *separately*: We worry only about the part of the sub-goal we can support, not about other parts that the same action may contradict. For $m = 1$, this means to ignore "delete lists" altogether as the same action never both supports and contradicts a single fact. For $m > 1$, we have to adequately (non-contradictingly) achieve all size-m sub-goals. That generalization is exactly the one made by $h^+(\Pi^{Cm})$. Definition 2 captures this by splitting up the goal (initially, the global goal fact set of the planning task) into all size-$\leq m$ sub-goals in the bottom case. For any given action in the middle case, these sub-goals are regressed separately, so each must be achieved non-contradictingly but contradictions across sub-goals are ignored.

There are two important subtleties in Definition 2, which distinguish it from what we would have to write in order to capture Π^C_{ce} instead of Π^C. First, we hand over the *union* $\bigcup_{g \in \mathcal{G}'} R(g,a)$ of the regressed sub-goals, forcing achievement of all these conditions conjunctively, like in Π^C when selecting C', for $a^{C'}$, to correspond to the set of conjunctions $\mathcal{G}'$. In particular, taking the union will give rise to *cross-dependencies* arising from several size-$\leq m$ sub-goals $g \in \mathcal{G}'$ ("cross-context π-fluents" in the parlance of Keyder et al. (2012)). To capture Π^C_{ce}, we can instead hand over each sub-goal $g \in \mathcal{G}'$ separately. Second, in the minimization, we minimize over pairs of action a and achieved sub-goal set $\mathcal{G}'$, as opposed to minimizing only over a and forcing $\mathcal{G}'$ to be maximal, i. e., setting $\mathcal{G}' = \{g \in \mathcal{G} \mid R(g,a) \neq \bot\}$. The latter would be suitable for capturing Π^C_{ce}, where there is no point in leaving out a "possible benefit" of the action a. In Π^C, that is not so because larger $\mathcal{G}'$ may give rise to additional cross-dependencies. For example, if $\mathit{eff}_a = \{p\}$ and $\mathcal{G} = \{\{p, q_1\}, \{p, q_2\}\}$ where q_1 and q_2 are impossible to achieve together, then $\mathcal{G}' = \{\{p, q_1\}, \{p, q_2\}\}$ leads to the unsolvable sub-goal $\{\{q_1, q_2\}\}$, while $\mathcal{G}' = \{\{p, q_1\}\}$ leads to the sub-goal $\{\{q_1\}, \{p, q_2\}\}$ which is solvable because we can achieve each of q_1 and $\{p, q_2\}$ separately.

To prove that Definition 2 does indeed capture $h^+(\Pi^{Cm})$, we start with the simple case $m = 1$, which will be employed below in the proof for the general case:

Theorem 1 $h^+ = h^{1+}$.

Proof: We show that, for $m = 1$, the h^{m+} equation simplifies to Equation 2. For $m = 1$, the bottom case just splits $\mathcal{G}$ up into its single goal facts. Hence the internal structure of $\mathcal{G}$ – the fact subsets it contains – does not matter; it matters only which facts are contained in any of these fact subsets. We can thus perceive $\mathcal{G}$ as a set goal facts, equivalently re-writing the equation to:

$$\begin{cases} 0 & \forall g \in \mathcal{G} : g \in I \\ 1 + \min_{a \in A, \emptyset \neq \mathcal{G}' \subseteq \{g \in \mathcal{G} \mid R(\{g\},a) \neq \bot\}} & \\ \quad h^{1+}((\mathcal{G} \setminus \mathcal{G}') \cup \bigcup_{g \in \mathcal{G}'} R(\{g\},a)) & \text{otherwise} \end{cases}$$

For a single goal fact $g \in \mathcal{G}'$, $R(\{g\}, a)$ is defined iff $g \in \mathit{eff}(a)$. Thus we can re-write the above to:

$$\begin{cases} 0 & \forall g \in \mathcal{G} : g \in I \\ 1 + \min_{a \in A, \emptyset \neq \mathcal{G}' \subseteq \mathcal{G} \cap \mathit{eff}(a)} & \\ \quad h^{1+}((\mathcal{G} \setminus \mathcal{G}') \cup \bigcup_{g \in \mathcal{G}'} R(\{g\},a)) & \text{otherwise} \end{cases}$$

Next, consider the regressed goal $(\mathcal{G}\backslash\mathcal{G}')\cup\bigcup_{g\in\mathcal{G}'} R(\{g\},a)$. For each $g\in\mathcal{G}'$, $R(\{g\},a)=pre(a)$. Thus the regressed goal is $(\mathcal{G}\setminus\mathcal{G}')\cup pre(a)$, giving us:

$$\begin{cases} 0 & \forall g\in\mathcal{G}: g\in I \\ 1+\min_{a\in A,\emptyset\neq\mathcal{G}'\subseteq\mathcal{G}\cap eff(a)} & \\ \quad h^{1+}((\mathcal{G}\setminus\mathcal{G}')\cup pre(a)) & \text{otherwise} \end{cases}$$

Observe that there is no point in choosing $\mathcal{G}'\subset\mathcal{G}\cap eff(a)$, i. e., using a to achieve a strict subset of its possible benefit $\mathcal{G}\cap eff(a)$, because that can only lead to a larger sub-goal $(\mathcal{G}\setminus\mathcal{G}')\cup pre(a)$. So we equivalently obtain:

$$\begin{cases} 0 & \forall g\in\mathcal{G}: g\in I \\ 1+\min_{a\in A,\emptyset\neq\mathcal{G}'=\mathcal{G}\cap eff(a)} & \\ \quad h^{1+}((\mathcal{G}\setminus\mathcal{G}')\cup pre(a)) & \text{otherwise} \end{cases}$$

With minimal re-writing, this turns into:

$$\begin{cases} 0 & \mathcal{G}\subseteq I \\ 1+\min_{a\in A,\emptyset\neq\mathcal{G}\cap eff(a)} & \\ \quad h^{1+}((\mathcal{G}\setminus eff(a))\cup pre(a)) & \text{otherwise} \end{cases}$$

This last equation is obviously equivalent to Equation 2, proving the claim. ∎

For $m=1$, $\Pi^{Cm}=\Pi$ so $h^+=h^+(\Pi^{Cm})$ and by Theorem 1 we get $h^+(\Pi^{Cm})=h^{m+}(\Pi)$ as desired. We now generalize this to arbitrary m:

Theorem 2 $h^+(\Pi^{Cm})=h^{m+}(\Pi)$.

Proof Sketch: By Theorem 1, for any Π we have $h^+(\Pi)=h^{1+}(\Pi)$. Applying this to $\Pi:=\Pi^{Cm}$, we get $h^+(\Pi^{Cm})=h^{1+}(\Pi^{Cm})$. It thus suffices to prove that $h^{1+}(\Pi^{Cm})=h^{m+}(\Pi)$. This is straightforward (but notationally cumbersome) based on comparing two equations, characterizing $h^{1+}(\Pi^{Cm})$ respectively $h^{m+}(\Pi)$.

For $h^{1+}(\Pi^{Cm})$, our equation (called *Equation I*) simply applies Definition 2 to Π^{Cm}:

$$\begin{cases} 0 & \forall g\in\mathcal{G}: g\subseteq I^C \\ 1+\min_{a^{C'}\in A^C,\emptyset\neq\mathcal{G}'\subseteq\{g\in\mathcal{G}|R(g,a^{C'})\neq\bot\}} & \\ \quad h^{m+}((\mathcal{G}\setminus\mathcal{G}')\cup\{\bigcup_{g\in\mathcal{G}'} R(g,a^{C'})\}) & \forall g\in\mathcal{G}: |g|\leq 1 \\ h^{m+}(\bigcup_{g\in\mathcal{G}}\{g'\subseteq g\mid |g'|=1\}) & \text{otherwise} \end{cases}$$

For $h^{m+}(\Pi)$, we need to do a little more work as we need to get rid of an irrelevant conceptual difference between Π^{Cm} and the equation defining h^{m+}: Whereas the latter splits up sub-goals only if their size is greater than m, Π^{Cm} always includes all possible π-fluents, even into sub-goals of size $\leq m$. Our new equation (called *Equation II*) modifies Definition 2 to do the same. We call $\mathcal{G}$ *completed* if, for all $g\in\mathcal{G}$, every $g'\subseteq g$ with $|g'|\leq m$ is contained in $\mathcal{G}$ as well:

$$\begin{cases} 0 & \forall g\in\mathcal{G}: g\subseteq I \\ 1+\min_{a\in A,\emptyset\neq\mathcal{G}'\subseteq\{g\in\mathcal{G}|R(g,a)\neq\bot\}} & \\ \quad h^{m+}((\mathcal{G}\setminus\mathcal{G}')\cup\{\bigcup_{g\in\mathcal{G}'} R(g,a)\}) & \\ \qquad \mathcal{G}\text{ is completed and } \forall g\in\mathcal{G}: |g|\leq m & \\ h^{m+}(\bigcup_{g\in\mathcal{G}}\{g'\subseteq g\mid |g'|\leq m\}) & \text{otherwise} \end{cases}$$

This is equivalent because we only add subsumed sub-goals.

Viewing each of Equations I and II as a tree whose root node is the "initializing call" containing the goal of the planning task, we show that the two trees are isomorphic. Namely, using the suffixes [I] and [II] to identify the tree, whenever the middle case applies we have:

$$(*)\ \mathcal{G}[I]=\{\{\pi_g\}\mid g\in\mathcal{G}[II]\}$$

To understand this intuitively, consider Equation II. This works on size-$\leq m$ sub-goals. Equation I works on singleton π-fluents representing size-$\leq m$ sub-goals. The original goal G gets split up into size-$\leq m$ subsets in II, vs. the π-fluents G^C in I, so we have (*). A sub-goal $g[I]=\pi_g$ in I can be regressed through $a^{C'}$ iff a achieves part of g and contradicts none of it; the same condition is applied in II. So the set $\mathcal{G}'$ of sub-goals tackled by a in II corresponds via (*) with that tackled by $a^{C'}$ in I. Finally, with $\mathcal{G}'$ having (*), the regressed sub-goal in I collects $pre(a)$ and $g\setminus eff(a)$ for all $\pi_g\in\mathcal{G}'$; the same is done in II, so (*) is preserved, concluding the proof. ∎

We remark that, from known results about $h^+(\Pi^{Cm})$, Theorem 2 implies that both $h^m\leq h^{m+}$ and $h^+\leq h^{m+}$: The marriage of h^m with h^+ yields a heuristic stronger than each of its sources, as one would expect.

Regarding dead-end detection power, it is easy to see that $h^{m+}=\infty$ iff $h^m=\infty$, i. e., like for $m=1$, the dead-end detection power of h^{m+} is the same as that of the corresponding critical-path heuristic.[3]

The above can be generalized to deal with arbitray C, i. e., to compute $h^{C+}=h^+(\Pi^C)$ using arbitrary Π^C instead of Π^{Cm}: In the bottom case of Definition 2, instead of splitting up into all size-m subsets, split up into the sets $c\in C$ (adapting the condition for the middle case accordingly to $\forall g\in\mathcal{G}\exists c\in C: g\subseteq c$).

Despite these niceties, we can't help but record:

Open Question 2 *What is this good for?*

We see two potential uses: (a) as a more direct way to *formulate* $h^+(\Pi^C)$ and thus ease leading proofs about its properties; and (b) as a more direct way to *compute* $h^+(\Pi^C)$, not necessitating a compilation step and thus being more efficient. Regarding (a), we haven't found any use case yet. Regarding (b), the most immediate idea is to extract "$h^{2\text{FF}}$" from a planning graph in a similar manner as for h^{FF} from a relaxed planning graph, considering pairs of sub-goal facts instead of single facts, following the correspondence between the equations characterizing h^{2+} vs. h^{1+}. However, there is no need for these equations to come up with $h^{2\text{FF}}$, and indeed Alcazar et al. (2013) already devised, implemented, and tested a variant of this idea, simply from the perspective of extending h^{FF} to correspond to Π^{C2}. As Alcazar et al. also already pointed out, "$h^{m\text{FF}}$" for arbitrary m can be computed from h^m respectively from an m-planning graph maintaining size-m mutexes. From that perspective, the main value of our work here is providing a theory background towards understanding and extending that technique.

[3] Similarly, for any C whose largest conjunction has size m, $h^+(\Pi^C)=\infty$ only if $h^m=\infty$.

It appears straightforward to extend $h^{m\text{FF}}$ to arbitrary conjunction sets C. A more tricky question, that might be answered using our formalization, is how exactly $h^{m\text{FF}}$ relates to the previous techniques $h^{\text{FF}}(\Pi^C)$ vs. $h^{\text{FF}}(\Pi^C_{ce})$.

Causal Graphs et al.

We now get back to the core motivation of this work, "scratching the itch". The aim is to understand under what circumstances "small" (i. e., polynomial-size) C is enough to render $h^+(\Pi^C)$ perfect. As an approach towards answering that question, we have taken the line of *identifying causal graph (CG) fragments (plus restrictions on the DTGs as needed) where $h^+(\Pi^{C2})$ is perfect*. In other words, adopt distinction lines as in many previous works on tractability, and see how far they carry when using only fact pairs.

The restriction to fact pairs is a bit arbitrary and mainly practically motivated. In particular, Keyder et al.'s (2012) implementation of semi-relaxed plan heuristics uses a subset of fact pairs. Then again, using all fact pairs in that implementation typically is infeasible, so we're still on the idealized side in our theory. In any case, the far more limiting fact here is that we got stopped in tracks right at the beginning. Having in mind initially to kill fork CGs quickly and then move on to more interesting quarters, we ended up spending lots of time racking our brains about even very small extensions to fork CGs, and indeed quite some time about fork CGs themselves.

What follows is thus a very simple fragment that we did manage to analyze. We remark that the proof is derived from a proof for (a generalization of) the VisitAll domain, for which also selecting all fact pairs is enough to render $h^+(\Pi^C)$ perfect.

We presume the reader is familiar with fork causal graphs. We will denote them here as planning tasks with a single "root variable" x, and with n "leaf" variables $y_1, \ldots, y_n$. The actions moving x do not have any preconditions on variables other than x, while the actions moving y_i may have preconditions on both y_i and x. So far, this is the standard fork CG setup. We impose the additional restriction, for all y_i, that y_i is Boolean and that there is only a single action affecting y_i. This essentially means that achieving the goal for y_i comes down to reaching a particular node in DTG_x (namely the one forming the precondition for y_i). We furthermore impose the restriction that x has no own goal, i. e., solving the task is just about moving the leaves into place (we will show later on that this restriction can be lifted, at least partially), and that every action moving x has a precondition on x.

We assume WLOG that initially each y_i is false, that the goal for each y_i is to be true (if y_i has no goal we can remove it without affecting either of h^* or $h^+(\Pi^C)$), and that the action for each y_i does have a precondition on x (else y_i moves independently and can be removed affecting h^* and $h^+(\Pi^C)$ in exactly the same way) and no precondition on y_i (that precondition could only be $y_i = False$, which is already true anyhow and thus affects neither h^* nor $h^+(\Pi^C)$).

We denote the DTG of x as a graph $DTG_x = (N, E)$ where the nodes N are the x values and the edges E correspond to the actions moving x. We denote by $n_i \in N$ the precondition on x of the action moving y_i, and by N_y the union of all n_i, i. e., those nodes we need to reach. We denote by n_0 the initial value of x. We denote facts $x = n$ simply by n, and we denote facts $y_i = True$ simply by y_i. We denote actions moving x by $go(d, d')$, and actions moving y_i by $do(i)$.

We refer to the class of planning tasks just described as *simple forks with binary leaves*.

Theorem 3 *$h^+(\Pi^{C2})$ is perfect for simple forks with binary leaves.*

The proof of this theorem is via a series of lemmas. First, it is easy to see that h^2-mutexes are recognized by Π^{C2}:

Lemma 1 *If $h^2(\{p, q\}) = \infty$, then $\pi_{p,q}$ is unreachable in Π^{C2}.*

This is simply because Π^{C2} captures support paths for all pairs of facts, just like h^2 does.

The following lemmas are basically concerned with paths, through the graph (N, E), that must be present in a relaxed plan for Π^{C2}. In the proofs, we will not explicitly distinguish the compiled actions in Π^{C2} from the original actions they are based on; instead, we will just talk about what preconditions are needed (will be present in Π^{C2}) if the original action a is to add a particular π-fluent π_c, i. e., if the corresponding conjunction c is added into the set C' for the compiled action $a^{C'}$.

The first lemma is a simple observation about achieving a pair of facts of the form "have y_i and now at n". Namely, to get that pair, we first need to get y_i and then move along a path to n:

Lemma 2 *Let Π be a simple fork with binary leaves, and let $\vec{a} = \langle a_1, \ldots, a_m \rangle$ be any sequence of actions applicable in Π^{C2}. Let s_k be the state that results from executing the prefix $\langle a_1, \ldots, a_k \rangle$. If $\pi_{y_i,n} \in s_k$, then $\langle a_1, \ldots, a_k \rangle$ contains a subsequence of actions that form a directed path in (N, E) from n_i to n.*

Proof: By induction on k. The base case, $k = 0$, is trivial, as it does not contain any $\pi_{y_i,n}$. Assume the claim holds for all $j < k$. The induction step proves it holds for k as well.

If $\pi_{y_i,n} \in s_k$, this either means that $\pi_{y_i,n} \in s_{k-1}$ (covered by induction hypothesis), or that a_k adds $\pi_{y_i,n}$. Say first that $a_k = do(i)$. Then $\pi_{y_i,n}$ can only be added if the compiled action has the precondition $\pi_{n_i,n}$ (only the y_i part can be added, n must have been true beforehand already). With Lemma 1, we must have $n_i = n$ or else the compiled action's precondition would be unreachable, in contradiction to applicability. But then, the directed path from n_i to n is empty and the claim holds trivially.

Say now that $a_k = go(d, d')$. Then $\pi_{y_i,n}$ can only be added if $a_k = go(d, n)$ (only the n part can be added, y_i must have been true beforehand already). But that compiled action has the precondition $\pi_{y_i,d}$, so by induction hypothesis $\langle a_1, \ldots, a_{k-1} \rangle$ contains a subsequence of actions that form a directed path from n_i to d in (N, E). Adding a_k to that subsequence forms the desired path from n_i to n. ∎

Our next lemma exploits the previous observation to show that any relaxed plan for Π^{C2} must, for every pair of the

target nodes N_y, contain a directed path between these two nodes in some order:[4]

Lemma 3 *Let Π be a simple fork with binary leaves, and let $\vec{a} = \langle a_1, \ldots, a_m \rangle$ be a relaxed plan for Π^{C2}. Then, for every $n_i \neq n_j \in N_y$, there is a subsequence of actions in $\vec{a}$ that form a directed path in (N, E) either from n_i to n_j or from n_j to n_i.*

Proof: The goal in Π^{C2} contains π_{y_i,y_j}. The only compiled actions which can achieve this are (1) $do(i)$ with precondition π_{y_j,n_i} (only the y_i part can be added, y_j must have been true beforehand already), or (2) $do(j)$ with precondition π_{y_i,n_j} (only the y_j part can be added, y_i must have been true beforehand already). Thus $\vec{a}$ must contain either of these two compiled actions. If $\vec{a}$ contains (i) $do(i)$ with precondition π_{y_j,n_i}, then by Lemma 2 $\vec{a}$ contains a subsequence of actions that form a directed path in (N, E) from n_j to n_i, showing the claim. Similarly for (2). ∎

We are now finally ready to prove Theorem 3 itself, by exploiting Lemma 3 in an argument as to how a relaxed plan can move through DTG_x:

Proof:[of Theorem 3] Let Π be a simple fork with binary leaves, and let $\vec{a} = \langle a_1, \ldots, a_m \rangle$ be a relaxed plan for Π^{C2}. It suffices to prove that there exists a subsequence of $\vec{a}$ that is a plan for Π.

By Lemma 3, for each pair of nodes $n_i, n_j \in N_y$, $\vec{a}$ contains a path from from n_i to n_j or vice versa. Hence, the graph

$$T_y = \langle N_y, \{(n_i, n_j) \mid \text{a path from } n_i \text{ to } n_j \text{ is in } \vec{a}\} \rangle$$

(or a subgraph of it, should $\vec{a}$ happen to contain paths in both directions between some pairs of nodes) is a tournament graph, and therefore must contain a Hamiltonian path, i.e., a directed path that visits every node (exactly once, in the graph T_y).[5] Furthermore, $\vec{a}$ must contain a path from n_0 to every n_i, as otherwise it could not achieve y_i. Hence, a subsequence of $\vec{a}$ must form a contiguous path, $n_0, n_{i_1}, \ldots, n_{i_n}$, through the nodes in N_y. Although in $\vec{a}$ the $do(i)$ actions can be applied at any time point after passing through n_i, whereas a plan for Π must apply $do(i)$ exactly when it is at n_i, the summed-up cost for applying all these actions is the same, proving the claim. ∎

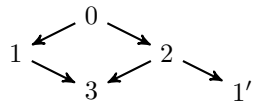
Having concluded this proof, the immediate question is whether all the restrictions on Π, such as the root variable having no own goal, and every action moving it having a precondition, are necessary.

[4]As a reminder: Being a relaxed plan for Π^{C2} is the same as being a plan for Π^{C2}, as we do not include any delete effects in our definition of the compilation here. We include the "relaxed" in the hope that being explicit is clearer.

[5]A *tournament graph* on n nodes is any directed graph obtained by assigning a direction to every edge in a complete undirected graph of size n. The proof that such graphs must contain a Hamiltonian path is due to Rédei (1934). A proof can be found in, e.g., the textbook by Moon (1968), or at `http://en.wikipedia.org/wiki/Tournament_(graph_theory)`.

Some of the restrictions can be relaxed. For example, if the root variable x has a goal value, n_x, and $n_x \neq n_i$ for all $n_i \in N_y$, the theorem still holds. The goal of Π^{C2} includes π_{y_i,n_x}, for all $n_i \in N_y$, so by Lemma 2 any relaxed plan $\vec{a}$ contains a path from n_i to n_x. Thus, adding n_x to the graph T_y in the proof above still leaves it a tournament graph, and because all edges between n_x and other nodes are (or can be chosen to be) directed towards n_x, this node can appear last in the Hamiltonian path.

The restriction to a single root variable "target node" per leaf variable (i.e., a single precondition $x = n_i$ common to actions that achieve $y_i = True$), on the other hand, is indeed necessary: if there are two options for achieving a fact y_i, we can construct a graph which allows relaxed plans in Π^{C2} to cheat, by achieving pairs π_{y_i,y_j} and π_{y_i,y_l} on separate branches of a directed tree. As a simple example, suppose there are three leaf variables, y_1, y_2 and y_3, and DTG_x is the following graph:



where 0 is the initial state of x, action $do(i)$ has precondition $x = i$, but there is an additional action $do'(1)$ with precondition $x = 1'$ (and effect $y_1 = True$). In Π^{C2}, the relaxed plan $go(0,1), do(1), go(1,3), do(3)$ achieves π_{y_1,y_3}, $go(0,2), do(2), go(2,3), do(3)$ achieves π_{y_2,y_3} and $go(0,2), do(2), go(2,1'), do'(1)$ achieves π_{y_1,y_2}, and hence the concatenation of all three achieves the goal. But the real problem has no solution.

Many of the other questions surrounding the restrictions in Theorem 3 do appear easy to answer, while a few may be hard. The reason we do not yet have answers is that most of our time was spent considering a slighly more general fragment than fork causal graphs, namely a more direct generalization of the VisitAll domain, where moving the root variable may have arbitrary side effects on subsets of Boolean leaf variables. This setting, it turned out, is substantially more complex to analyze.

In any case, the real open question remains:

Open Question 3 *Can the approach of analyzing CG-based tractable fragments ever be brought up to a level suitable for targeted selection of C in practice?*

Our idea here is, if for sufficiently large/many fragments of planning we know exactly which C are needed to render $h^+(\Pi^C)$ perfect, then we may be able to use these as "building blocks" for selection methods with a strong theory justification. In particular, for fact-pair cases, the idea would be to not necessarily consider all fact pairs but just the minimal subsets required.

Our speed of progress so far, and counter-examples identified for very simple fragments of planning, suggests that this approach is challenging to say the least, doomed perhaps. But the jury is still out. We speculate some more in the conclusion, and for now get back to some actual results:

$h^+(\Pi^C)$ vs. Red-Black vs. Fluent Merging

A different way to approach the power of $h^+(\Pi^C)$ is to compare it with other partial delete relaxation methods, i. e., alternative methods able to interpolate all the way between h^+ and h^*. Exactly two such alternative methods are known, at this time: *red-black planning* (Katz, Hoffmann, and Domshlak 2013b; 2013a; Katz and Hoffmann 2013) and *fluent merging* (van den Briel, Kambhampati, and Vossen 2007; Seipp and Helmert 2011). *Which of these approaches can simulate which other ones, i. e., compute an at least as good heuristic, with polynomial overhead?*

As fluent merging necessitates the use of 0-cost actions, in what follows we consider the arbitrary-costs case. The answers to the question we are posing are the same anyhow when assuming uniform costs (for those cases where that assumption is possible).

In fluent merging, we choose a subset $M \subseteq V$ of variables to merge, and replace them with a single variable v^M whose domain is the cross-product of the domains of $v \in M$.[6] Every action a that touches any $v \in M$ is then replaced by a set of actions resulting from the enumeration of possible precondition/effect values of v^M (i. e., states over M) that match a's precondition and effect. That is, we complete $\mathit{eff}(a)$ with an assignment p to the remaining variables, where p matches $pre(a)$; we complete $pre(a)$ with the same assignment p on variables not occuring in $\mathit{eff}(a)$, and with an arbitrary assignment on those variables that do occur in $\mathit{eff}(a)$. If M touches the goal, then an artificial goal value is introduced for v^M, reached by an artificial 0-cost action from every assignment to M that complies with the goal. Denote the resulting heuristic, i. e., the length of an optimal relaxed plan in the pre-merged task, by $h^{\text{Merge}+}$. It hasn't to our knowledge been noted before, but is obvious, that for $M = V$ we get $h^{\text{Merge}+} = h^*$ (so indeed this is an "interpolation method" in the sense above).

In red-black planning, we "delete-relax" only a subset V^R of the finite-domain state variables (the "red" ones), applying the original semantics to the remaining variables V^B (the "black" ones). That is, red variables accumulate their values ($\mathit{eff}(a)$ gets added to the state) while the black variables switch between their values ($\mathit{eff}(a)$ over-writes the state). Denote the resulting heuristic, i. e., the length of an optimal red-black plan, by $h^{\text{RB}+}$. Obviously, setting $V^B = \emptyset$ we get $h^{\text{RB}+} = h^+$, and setting $V^R = \emptyset$ we get $h^{\text{RB}+} = h^*$. The known "tractable fragments" (i. e., polynomial-time satisficing red-black plan generation) require (a) a fixed number of black variables each with fixed domain size, or (b) an acyclic *black causal graph* (projection of the causal graph onto V^B) where each black variable has only invertible value transitions in its DTG.

It is not completely clear what "polynomial overhead" should be taken to mean in this context. We choose to ignore the complexity of optimal partially-relaxed-plan generation, which makes sense as this underlies all three frameworks and will be approximated by satisficing partially-relaxed-plan generation just like in the standard delete relaxation. Given this, "polynomial overhead" for fluent merging means that $|M|$ is fixed; for $h^+(\Pi^C)$ we take it to mean that $|C|$ is polynomially bounded.[7] For red-black planning, we take "polynomial overhead" to mean "inside a known tractable fragment"; this is not fair as there may be yet unknown tractable fragments, but it is the best we can do for now.

As per this simulation framework, it turns out that all three approaches are orthogonal, with a single exception:

Theorem 4 *None of $h^+(\Pi^C)$, red-black planning, and fluent merging can simulate any other with polynomial overhead, except that $h^+(\Pi^C)$ simulates fluent merging on M when setting C to contain all fact conjunctions c over M (including c mentioning the same variable more than once).*

Proof Sketch: To see that red-black planning cannot simulate either of $h^+(\Pi^C)$ or fluent merging, it suffices to construct an example whose only "flaw" is small and easy to fix, but outside a known tractable fragment. This can be based, e. g., on having to buy a car, consuming a piece of gold, but the goal being to have both the car and the gold. Merging the two variables (car and gold) yields $h^{\text{Merge}+} = h^* = \infty$, and a single conjunction yields $h^+(\Pi^C) = \infty$. For $h^{\text{RB}+} = \infty$ we would need both variables to be black, yielding a cyclic causal graph; it is easy to scale variable domains and the number of variables so that neither $h^{\text{Merge}+} = h^*$ nor $h^+(\Pi^C) = h^*$ is affected.

$h^+(\Pi^C)$ cannot simulate red-black planning because there are planning tasks whose causal graphs are lines (in particular, DAGs) and all of whose variables are invertible, but where h^* is exponentially large. This is tractable for $h^{\text{RB}+} = h^*$ (using a succinct plan representation), but is not tractable for $h^+(\Pi^C)$ by Proposition 2.

Fluent merging cannot simulate red-black planning because sometimes painting a single variable black suffices whereas we would need to merge all variables to obtain $h^{\text{Merge}+} = h^*$. One such example is "star-shaped switches", where a robot starts in the middle node of a star graph, has to move to every leaf node turning on a switch, and has to be back in the middle at the end. Painting the robot variable black obviously gives $h^{\text{RB}+} = h^*$. However, if M leaves out a single variable then $h^{\text{Merge}+} < h^*$. This is obvious for the robot variable. If switch variable $v \notin M$, then a relaxed plan can solve v_M as appropriate (switching all other switches on and moving back to the middle), then move outwards to v's node and switch it on, but not move back to the middle as the two goals "other-switches-on-and-robot-at-middle" as well as "switch-v-on" are both already true.

The same example shows that fluent merging cannot simulate $h^+(\Pi^C)$, as by Theorem 3 we have $h^+(\Pi^{C2}) = h^*$.

Consider finally the only positive result. Denoting the pre-merged task by Π^M, our proof considers an optimal

[6] Note that this is a restricted version of the technique, merging only a single subset of variables. One can instead merge several subsets, potentially with overlaps between them. We restrict outselves to the simpler variant in what follows.

[7] We ignore the exponential growth of Π^C in $|C|$ because (a) this can be largely fixed using Π^C_{ce} (Keyder, Hoffmann, and Haslum 2012), and (b) it appears that alternate methods for computing approximations of $h^+(\Pi^C)$, not going via a compilation, can avoid that blow-up altogether, cf. our comments below Open Question 2.

relaxed plan $\vec{a} = \langle a_1^{C'_1}, \ldots, a_n^{C'_n} \rangle$ for Π^C, and shows that we can transform $\vec{a}$ step-by-step into a relaxed plan $\vec{a}^M = \langle a_1^M, \ldots, a_n^M \rangle$ for Π^M based on the same actions a_i from the original planning task. Then $\vec{a}^M$ has the same cost as $\vec{a}$, implying that $h^+(\Pi^M) \leq h^+(\Pi^C)$ as we need to prove.

While that idea sounds simple, spelling it out was unexpectedly cumbersome, taking us a few iterations and ending up being a full page long in this format (perhaps there are simpler proofs). Omitting the details, consider the structure of relaxed plans in Π^C vs. Π^M. In the latter, the M-states visited form a tree, the root being the initial state, actions a_i^M connecting to any M-state already visited. But what is the structure in Π^C? While it is easy to see that each C'_i contains at most one full assignment to M (otherwise the precondition would contain an unreachable mutex pair, of different values for the same variable), nothing forces the C'_i to include any full assignment. So the structure of $\vec{a}$ is less rigid as that of relaxed plans in Π^M, making commitments only where necessary. Our proof shows how to instantiate these partial commitments to full commitments, extending each C'_i to correspond exactly to a full assignment to M: At the initial state, the commitment is full already. Every action i must have a preceding action $r(i)$ adding the π-fluent corresponding to $a_i^{C'_i}$'s entire set of precondition facts. Assuming that $C'_{r(i)}$ has been fully committed already, we can extend the partial commitment made by C'_i accordingly. ∎

We would like to note that, for once, there are no open questions left here. Except of course what would happen were we to identify further tractable fragments of red-black planning, and whether more general variants of fluent merging (several variable subsets with potential overlaps) make a difference. As that is not "open enough" for the spirit of this paper, we now get into open questions for real:

Some Wild Speculations (aka "Conclusion")

Disregarding issues such as exponential separations between the different variants of Π^C, or the role of mutex pruning (removing actions with known mutexes in the precondition) in all this, let us focus our speculation on the questions we started out with: Can we characterize more accurately the circumstances under which $h^+(\Pi^C)$ becomes perfect? When does that require C to be exponentially large, and when is polynomial-size C enough?

Our most direct answer to these questions is Theorem 3, tractability of a restricted fork fragment. While our original plan had been to extend that result "bottom-up", proving tractability of different/ever larger fragments, the effort it took to analyze even slightly larger fragments suggests that perhaps a "top-down" approach might be more suitable:

Open Question 4 *Can we identify easily testable sufficient conditions for some subset D of conjunctions to* not *be required for rendering $h^+(\Pi^C)$ perfect?*

For example, if there is no undirected CG path between two variables, then presumably we do not need to include conjunctions involving both. But what if there is no *directed* CG path between them?

Or perhaps we can make progress by considering particular benchmark domains:

Open Question 5 *In which IPC benchmarks is h^{2+}, i. e., $h^+(\Pi^C)$ with C being all fact pairs, perfect? In those domains where it isn't, how does the search topology (local minima, exit distances, unrecognized dead-ends) differ from that known for h^+ (Hoffmann 2005)?*

A major source of speculations is whether we can somehow identify structural criteria – based on whatever notions, not necessarily efficiently testable – under which $h^+(\Pi^C)$ becomes perfect. Let us start with the most plausible one:

Open Question 6 *Say that a task Π is "m-decomposable" if there exists a partitioning $\{V_i\}$ of its variables whose largest V_i has size m, and where the length of an optimal plan for Π is equal to the sum of lengths of optimal plans for the projections onto V_i. Is h^{m+} perfect?*

While that is more like a conjecture than an open question, its practical use is doubtful: If we have m-decomposability, then basically we can split up the planning task into its pieces and have no need for a global heuristic addressing the whole task. Unless, of course, we don't actually know what the decomposition is, only its size; but that seems unlikely to happen in practice (?)

We close the paper with what are probably our two most speculative open questions. One is basically an attempt to answer Open Question 2:

Open Question 7 *Can syntactical criteria be identified which imply that the h^{m+} equation simplifies to Equation 1?*

We haven't got any idea how to do this; it should be said though that we did not spend much time trying.

Finally, thinking about the size of conjunctions needed often corresponds to thinking about "the value of how many variables we need to remember in order to avoid cheating". The "remembering" here corresponds to deleted values that are required again later on. The number of variables affected in this way intuitively corresponds to a "level of interference". For example, in VisitAll (and other fork-like domains) the only variable whose value we need to remember is the robot (the root of the fork); in puzzles, by contrast, achieving a desired value may typically involve deleting arbitrarily many other desired values. Taking "level of interference" m to be the maximal number of variables affected while achieving a target value, plus one for the target variable itself, the question is:

Open Question 8 *Say that a task Π has level of interference m (whatever that means, exactly). Is h^{m+} perfect?*

We are looking forward to some answers in the future, apologize for posing so many un-answered questions, and thank you for not having laid the paper aside before reaching this sentence.

Acknowledgments. We thank the anonymous reviewer (aka Malte Helmert) for many detailed and helpful comments.

References

Alcázar, V.; Borrajo, D.; Fernández, S.; and Fuentetaja, R. 2013. Revisiting regression in planning. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*, 2254–2260. AAAI Press.

Haslum, P. 2009. $h^m(P) = h^1(P^m)$: Alternative characterisations of the generalisation from $h^{\max}$ to h^m. In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, 354–357. AAAI Press.

Haslum, P. 2012. Incremental lower bounds for additive cost planning problems. In Bonet, B.; McCluskey, L.; Silva, J. R.; and Williams, B., eds., *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, 74–82. AAAI Press.

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

Moon, J. W. 1968. *Topics on Tournaments*. Holt, Rinehart and Winston, Inc. `http://www.gutenberg.org/ebooks/42833`.

Rédei, L. 1934. Ein kombinatorischer satz. *Acta. Litt. Szeged* 7:39–43.

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.; Kambhampati, S.; and Vossen, T. 2007. Fluent merging: A general technique to improve reachability heuristics and factored planning. In *ICAPS 2007 Workshop on Heuristics for Domain-Independent Planning: Progress, Ideas, Limitations, Challenges*.