LP Heuristics over Conjunctions: Compilation, Convergence, Nogood Learning

Marcel Steinmetz and **Jörg Hoffmann**
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
{steinmetz,hoffmann}@cs.uni-saarland.de

Abstract

Two strands of research in classical planning are LP heuristics and conjunctions to improve approximations. Combinations of the two have also been explored. Here, we focus on *convergence* properties, forcing the LP heuristic to equal the perfect heuristic h^* in the limit. We show that, under reasonable assumptions, *partial variable merges* are strictly dominated by the compilation $\Pi^{\mathcal{C}}$ of *explicit conjunctions*, and that both render the *state equation heuristic* equal to h^* for a suitable set C of conjunctions. We show that consistent *potential heuristics* can be computed from a variant of $\Pi^{\mathcal{C}}$, and that such heuristics can represent h^* for suitable $\mathcal{C}$. As an application of these convergence properties, we consider sound nogood learning in state space search, via refining the set C. We design a suitable refinement method to this end. Experiments on IPC benchmarks show significant performance improvements in several domains.

1 Introduction

In classical planning, LP heuristics approximate cost-to-goal through linear constraints. The *state equation heuristic* [van den Briel *et al.*, 2007a; Bonet, 2013] formulates LP constraints over the number of action occurrences needed. Apart from integrating additional sources of information [Bonet and van den Briel, 2014; Pommerening *et al.*, 2014], a successful approach are *potential heuristics* [Pommerening *et al.*, 2015; Seipp *et al.*, 2015], based on the dual of the state equation. The solution of that dual LP defines weights for variable values. Computed just once on the initial state, these weights define a consistent heuristic function for the entire state space.

Another research line uses variable-value (fact) conjunctions to improve approximations [van den Briel *et al.*, 2007b]. The compilation $\Pi^{\mathcal{C}}$ renders a set $\mathcal{C}$ of conjunctions *explicit*, forcing delete relaxed plans to *converge* to real plans in the limit, i. e., for suitable $\mathcal{C}$ [Haslum, 2009; 2012; Keyder *et al.*, 2014]. This leads to powerful heuristics, and in particular enables sound forward-search *nogood learning*, where the heuristic is iteratively *refined* based on the dead-end states encountered [Steinmetz and Hoffmann, 2017].

Here, we consider LP heuristics over conjunctions. This is not novel per se: *partial variable merges* enhance the state equation heuristic by constraints over conjunctions [Bonet and van den Briel, 2014]; potential heuristics can be defined over conjunctions as well [Seipp *et al.*, 2016a; Pommerening *et al.*, 2017]. We contribute new results pertaining to the use of the $\Pi^{\mathcal{C}}$ compilation, to convergence properties, and to the exploitation of convergence for nogood learning.

Seipp et al. [2016a] examined the size of conjunctions needed for a potential heuristic to find a plan without search. Pommerening et al. [2017] designed potential heuristics over arbitrary sets of conjunctions. While they provide an efficient construction method for potential heuristics over fact pairs, they also showed that the construction over fact triples is already computationally hard. Here, we show that for tasks in transition normal form (TNF) [Pommerening and Helmert, 2015], partial variable merges are strictly dominated by the compilation $\Pi^{\mathcal{C}}$, and that both render the state equation heuristic equal to h^* for suitable $\mathcal{C}$. We show that consistent potential heuristics can be constructed from $\Pi^{\mathcal{C}}$.[1] We show that, together with a (trivial to compute) upper bound $U^* \in \mathbb{R}$ on $h^*(s)$ for solvable s, one can choose $\mathcal{C}$ so that the $\Pi^{\mathcal{C}}$ potential heuristic equals h^*.

We exploit these properties, motivated by the success of LP heuristics in the Unsolvability-IPC [Seipp *et al.*, 2016b], in forward-search nogood learning. Convergence is essential here: Nogood learning using a heuristic h requires a *refinement method* that, given a dead-end state s not pruned by h so far, guarantees to refine h into h' that prunes s. We design such a refinement method for LP heuristics, choosing new conjunctions suitable for the state equation and hence also the potential heuristic. Experiments on IPC benchmarks show significant performance improvements in several domains.

Throughout the paper, we omit many technical details. We refer the reader to the TR [Steinmetz and Hoffmann, 2018].

2 Preliminaries

We consider planning tasks in FDR notation [Bäckström and Nebel, 1995; Helmert, 2009]. A *task* is a tuple $\Pi =$

[1] This does not contradict Pommerening et al.'s [2017] hardness result as $\Pi^{\mathcal{C}}$ may grow exponentially in $|\mathcal{C}|$. From this perspective, our result can be viewed as a way of identifying feasible potential heuristics over conjunctions: where $\Pi^{\mathcal{C}}$ is small.

$\langle \mathcal{V}, \mathcal{A}, s_\mathcal{I}, s_* \rangle$. $\mathcal{V}$ is a set of *variables* v, each associated with a finite domain $\mathcal{D}_v$. A *state* is a complete assignment to V. $s_\mathcal{I}$ is the *initial state*. The *goal* s_* is a partial variable assignment. $\mathcal{A}$ is a finite set of *actions*. Each action $a \in \mathcal{A}$ is associated with a *precondition* pre_a and an *effect* eff_a, both partial variable assignments, and a non-negative *cost* $\mathsf{cost}_a \in \mathbb{R}_0^+$. We assume that $pre_a(v) \neq \mathit{eff}_a(v)$ when both are defined.

A *fact* is a variable-value pair $p = \langle v, d_v \rangle$. For partial variable assignments P, we denote by $\mathcal{V}(P)$ the set of variables for which P is defined. For variables $v \notin \mathcal{V}(P)$, we also write $P(v) = \bot$. We often treat (partial) variable assignments P as sets of facts $\{\langle v, P(v) \rangle \mid v \in \mathcal{V}(P)\}$. We say that two assignments P and P' are *compatible*, written $P||P'$, if for all $v \in \mathcal{V}(P) \cap \mathcal{V}(P')$ we have $P(v) = P'(v)$.

An action a is *applicable* in a state s if $pre_a \subseteq s$, and the application results in the state $s[\![a]\!]$ where $s[\![a]\!](v) = \mathit{eff}_a(v)$ if $v \in \mathcal{V}(\mathit{eff}_a)$ and $s[\![a]\!](v) = s(v)$ otherwise. Action sequences $\pi = \langle a_1, \ldots, a_n \rangle$ are applied iteratively, and the outcome state is denoted $s[\![\pi]\!]$. If $s_* \subseteq s[\![\pi]\!]$, then π is a *plan* for s. A plan for Π, or just plan, is a plan for $s_\mathcal{I}$. The cost of π is $\sum_{i=1}^{n} \mathsf{cost}_{a_i}$. A plan is *optimal* if its cost is minimal among all plans. If there is no plan for s, then s is a *dead-end*. If $s_\mathcal{I}$ is a dead-end, we say that Π is *unsolvable*.

The set of all states in Π is denoted $\mathcal{S}$. A *heuristic* is a function $\mathcal{S} \mapsto \mathbb{R}_0^+ \cup \{\infty\}$. The *perfect heuristic* h^* assigns each state s the cost of an optimal plan for s, or ∞ if s is a dead-end. A heuristic h is *admissible* if $h(s) \leq h^*(s)$ for all $s \in \mathcal{S}$; h is *consistent* if, whenever $a \in \mathcal{A}$ is applicable in s, we have $h(s) \leq h(s[\![a]\!]) + \mathsf{cost}_a$.

A notation for *regression*, defined in the usual way, will be convenient. The regression of P over a is $\mathsf{regr}(P, a) = (P \setminus \mathit{eff}_a) \cup pre_a$ if $\mathit{eff}_a \cap P \neq \emptyset$ and $\mathit{eff}_a || P$ and $(P \setminus \mathit{eff}_a) || pre_a$; otherwise, $\mathsf{regr}(P, a) = \bot$.

We will sometimes consider *transition normal form (TNF)* [Pommerening *et al.*, 2015]. This imposes that (TNF1) $\mathcal{V}(\mathit{eff}_a) \subseteq \mathcal{V}(pre_a)$ for all $a \in \mathcal{A}$,[2] and (TNF2) $\mathcal{V}(s_*) = \mathcal{V}$. Every task can be transformed into TNF in polynomial time.

We next introduce the $\Pi^\mathcal{C}$ compilation. We follow Haslum [2012], with small modifications suiting our context.

A *conjunction* c is a partial variable assignment. To represent a set $\mathcal{C}$ of conjunctions explicitly in a given task Π, the $\Pi^\mathcal{C}$ compilation introduces a new Boolean variable π_c for each $c \in \mathcal{C}$; abusing notation, we identify π_c with the fact $\langle \pi_c, 1 \rangle$. For a partial assignment P, $P^\mathcal{C} := P \cup \{\langle \pi_c, 1 \rangle \mid c \in \mathcal{C}, c \subseteq P\} \cup \{\langle \pi_c, 0 \rangle \mid c \in \mathcal{C}, c \not\parallel P\}$ augments P with the conjunctions it contains as well as the negation of the conjunctions it conflicts with. A set of conjunctions $C \subseteq \mathcal{C}$ is *compatible* if all $c, c' \in C$ are pairwise compatible.

Definition 1. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_\mathcal{I}, s_* \rangle$ be a task, and $\mathcal{C}$ be a set of conjunctions. Then $\Pi^\mathcal{C} := \langle \mathcal{V}^\mathcal{C}, \mathcal{A}^\mathcal{C}, s_\mathcal{I}^\mathcal{C}, s_*^\mathcal{C} \rangle$ where $\mathcal{V}^\mathcal{C} = \mathcal{V} \cup \{\pi_c \mid c \in C\}$ and $\mathcal{A}^\mathcal{C}$ contains an action a^C for every pair $a \in \mathcal{A}$ and compatible $C \subseteq \mathcal{C}$ such that: (1) for all $c \in C$, $\mathsf{regr}(c, a) \neq \bot$, and (2) for every $c' \in \mathcal{C}$, if $\mathsf{regr}(c', a) \neq \bot$ and $\mathsf{regr}(c', a) \subseteq (pre_a \cup \bigcup_{c \in C} \mathsf{regr}(c, a))$, then $c' \in C$. The action a^C is given by (i) $pre_{a^C} = [pre_a \cup \bigcup_{c \in C} \mathsf{regr}(c, a)]^\mathcal{C}$, (ii) $\mathit{eff}_{a^C} = \mathit{eff}_a \cup \{\langle \pi_c, 1 \rangle \mid c \in C\} \cup \{\langle \pi_{c'}, 0 \rangle \mid c' \in \mathcal{C}, c' || pre_{a^C}, c' \not\parallel \mathit{eff}_a\}$, and (iii) $\mathsf{cost}_{a^C} = \mathsf{cost}_a$.*

Intuitively, a^C represents an *occurrence* of a that makes all $c \in C$ true. For this to happen, the regression of each c over a must be true beforehand, as per (i). As per (ii), a conjunction c' potentially invalidated by a is false afterwards. Condition (2) assures consistency: if an occurrence a^C always makes true a conjunction c, then π_c must be set to true necessarily.

With the possible C being subsets of $\mathcal{C}$, $|\mathcal{A}^\mathcal{C}|$ may grow exponentially in $|\mathcal{C}|$. This can be ameliorated (but not overcome entirely) with mutex information [Keyder *et al.*, 2014]. Compatibility of C as postulated here is a special case thereof.

Plan equivalence between Π and $\Pi^\mathcal{C}$ can be easily shown by adapting Haslum's [2012] proof to our slightly modified Π^C definition. An extension of this equivalence result to individual transitions will become handy later on:

Lemma 1. *For every Π and every $\mathcal{C}$, it holds that every consistent heuristic h for Π is consistent in $\Pi^\mathcal{C}$, and vice versa every consistent heuristic h for $\Pi^\mathcal{C}$ is consistent in Π.*

Proof (sketch). Since every transition in $\Pi^\mathcal{C}$ has a corresponding transition in Π, the first direction is easy. For the other direction, the definition of $\Pi^\mathcal{C}$ ensures, for every state s and every applicable action a, that there exists an action occurrence a^C of a such that $(s[\![a]\!])^\mathcal{C} = s^\mathcal{C}[\![a^C]\!]$. Hence, consistency in $\Pi^\mathcal{C}$ implies also consistency in Π. □

3 The State Equation Heuristic

We introduce the state equation heuristic, and discuss its extensions to deal with conjunctions.

3.1 Definitions

The *state equation* (SEQ) describes a relation between variable-value changes, the *net-changes*, that every plan must satisfy. A fact $p = \langle v, d_v \rangle$ is *produced* by an action a if $\mathit{eff}_a(v) = d_v$; p is *consumed* by a if $pre_a(v) = d_v$ and $v \in \mathcal{V}(\mathit{eff}_a)$. Let s be any state, π be any plan for s, and p be any fact. Every consumption of p along π requires its production beforehand. If p is true in s, then p can be consumed once more than it is produced. If p must be true after the application of π, then p must be produced more often than it is consumed. So, let Count_a^π be the number of occurrences of action a in π. Denote by $\mathit{Prod}(p)$ ($\mathit{Cons}(p)$) the set of all actions that produce (consume) p. Then the SEQ for p is

$$\sum_{a \in \mathit{Prod}(p)} \mathsf{Count}_a^\pi - \sum_{a \in \mathit{Cons}(p)} \mathsf{Count}_a^\pi \geq \Delta_p(s) \qquad (1)$$

where $\Delta_p(s) = 1$ if $p \notin s$ and $p \in s_*$; $\Delta_p(s) = -1$ if $p \in s$ and $p \notin s_*$; and $\Delta_p(s) = 0$ otherwise.

The *state equation heuristic* h^{SEQ} is defined via an LP. The LP contains one variable $\mathsf{Count}_a \in \mathbb{R}_0^+$ for every action a. For every fact p the LP contains the constraint given by Equation (1) for p, choosing the right hand side $\Delta_p(s)$ according to the state s for which $h^{\mathrm{SEQ}}(s)$ is being computed. The objective function is to minimize $\sum_{a \in \mathcal{A}} \mathsf{Count}_a \cdot \mathsf{cost}_a$. If the LP has an optimal solution, then $h^{\mathrm{SEQ}}(s)$ gives the respective

[2] This differs slightly from the TNF definition in literature where $\mathcal{V}(pre_a) = \mathcal{V}(\mathit{eff}_a)$ is required. We do so for simplicity only. All our results apply directly to the original version as well.

the objective value. Otherwise $h^{\text{SEQ}}(s) = \infty$. As every plan must satisfy Equation (1) for every p, h^{SEQ} is admissible.

An important weakness of the state equation heuristic is that *prevail* conditions of actions, i. e., preconditions $pre_a(v)$ where $v \notin \mathcal{V}(\mathit{eff}_a)$, are disregarded completely.

Example 1. *Consider the transportation example in Figure 1.*

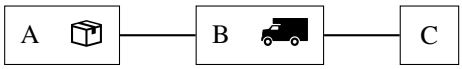

Figure 1: Initial state in the transportation example.

There are two variables t and p, indicating the position of the truck respectively package. The truck can move freely on the depicted map through $move$ actions. The package can be $loaded$ or $unloaded$ at the current truck position. All actions have cost 1. The initial state is given by $s_{\mathcal{I}} = \{\langle t = B\rangle, \langle p = A\rangle\}$, the goal is $s_* = \{\langle p = C\rangle\}$. The h^{SEQ} value for this state is 2, accounting only for loading and unloading the package. Truck movements are not counted since loading and unloading the package prevail the truck position.

3.2 The State Equation over Conjunctions

The weakness just discussed can be addressed by considering net-changes over conjunctions instead of single facts.

Example 2. *Consider the set of conjunctions $\mathcal{C} = \{c_1, c_2\}$ for $c_1 = \{\langle t = A\rangle, \langle p = A\rangle\}$ and $c_2 = \{\langle t = C\rangle, \langle p = T\rangle\}$. Loading the package at A now requires and consumes π_{c_1}, and unloading the package at C consumes π_{c_2}. To produce π_{c_1}, the truck has to move to location A. To produce π_{c_2}, the truck has to move to C. Further, to satisfy Equation (1) for $t = B$, the truck needs to move back from A to B. This results in the perfect heuristic value 5; indeed, $h^{\mathit{SEQ}}[\Pi^{\mathcal{C}}](s_{\mathcal{I}}) = 5$.*

Bonet and van den Briel [2014] designed *partial variable merges* to extend h^{SEQ} to conjunctions. We now compare this technique to the computation of h^{SEQ} in $\Pi^{\mathcal{C}}$, and we show convergence of both to h^* for Π in TNF. For space reasons, we give a summary only. The TR contains the details.

Like the $\Pi^{\mathcal{C}}$-compilation, partial variable merges consider a set of conjunctions $\mathcal{C}$. But conjunctions $c, c' \in \mathcal{C}$ are put into relation only when they instantiate the same variables, $\mathcal{V}(c) = \mathcal{V}(c')$. Hence the name "partial variable merges": Bonet and van den Briel start from the simpler idea of pre-merging entire variable subsets $V \subseteq \mathcal{V}$, extending Π by a new variable representing this product; they improve over that idea by considering only particular value tuples within the product. As a result, their LP encoding grows polynomially in $|\mathcal{C}|$, but might lose information relative to $\Pi^{\mathcal{C}}$ because no constraints are included *across* (conjunctions over) different variable subsets V, V'.

Specifically, partial variable merges are based on notions of *potential* producers and consumers, i. e., actions whose applications *can* achieve respectively invalidate c: $PProd(c) = \{a \in \mathcal{A} \mid \mathsf{regr}(c, a) \neq \bot\}$ and $PCons(c) = \{a \in \mathcal{A} \mid c \not\parallel \mathit{eff}_a, c || pre_a\}$. This complication arises because the impact of an action a on a conjunction c depends on the context in which a is applied: on the action occurrence.

While $\Pi^{\mathcal{C}}$ enumerates possible action occurrences, variable merges handle each subset of $\mathcal{C}$ sharing the same variables V separately. Denote by $\Pi|_V$ the product of V (which corresponds to the projection of Π onto V). To represent those $\Pi|_V$ states P where $P \notin \mathcal{C}$, an abstract state $\top$ is introduced. The transitions within $\Pi|_V$ are abstracted by inserting $\top$ whenever the start or end state of a transition is not contained in $\mathcal{C}$. Equation (1) for a conjunction c is then defined by summing over $a \in PProd(c)$ respectively $a \in PCons(c)$, with a separate occurrence-counter variable $\mathsf{Count}_a^{x \to x'}$ for every abstract transition $x \to x'$ induced by a. Finally, these separate counters are related back to the main action counters by adding the constraint that the sum over $\mathsf{Count}_a^{x \to x'}$ is $\leq \mathsf{Count}_a$. We denote the resulting heuristic by $h^{\mathcal{C}\text{SEQ}}$.

Theorem 1. *For every Π in TNF, every set of conjunctions $\mathcal{C}$, and every state s, it holds that $h^{\mathit{SEQ}}[\Pi^{\mathcal{C}}](s) \geq h^{\mathcal{C}\mathit{SEQ}}(s)$.*

Proof (sketch). Let $\text{Seq}[\Pi^{\mathcal{C}}]$ be the LP underlying $h^{\text{SEQ}}[\Pi^{\mathcal{C}}](s)$, and $\text{Seq}\mathcal{C}$ that underlying $h^{\mathcal{C}\text{SEQ}}(s)$. Every solution to $\text{Seq}[\Pi^{\mathcal{C}}]$ can be transformed into a solution to $\text{Seq}\mathcal{C}$, with equal objective value. The proof is technical but straightforward. □

Theorem 2. *There exists families of Π and $\mathcal{C}$ s.t., to obtain $h^{\mathcal{C}'\text{SEQ}} \geq h^{\mathit{SEQ}}[\Pi^{\mathcal{C}}]$, $\mathcal{C}'$ must be exponentially larger than $\mathcal{C}$.*

Proof (sketch). This happens, e. g., in a transportation example where n packages must be transported from B to A, and truck-load capacity is 1. In $h^{\text{SEQ}}[\Pi^{\mathcal{C}}]$, considering all conjunctions of size up to 3 makes visible that no two packages can be in the truck at the same time, yielding $h^{\text{SEQ}}[\Pi^{\mathcal{C}}] = h^*$. The partial variable merges in $h^{\mathcal{C}\text{SEQ}}$, however, cannot account perfectly for the interactions across packages unless all of them are considered jointly in the same $\Pi|_V$. □

For general tasks Π, the relation between $h^{\text{SEQ}}[\Pi^{\mathcal{C}}]$ and $h^{\mathcal{C}\text{SEQ}}$ is however not so clear anymore. Complications stem from actions a affecting variables v without precondition on v. $\Pi^{\mathcal{C}}$ cannot relate the consumption of any conjunction c where $v \in \mathcal{V}(c)$ with a's action occurences. In contrast, $h^{\mathcal{C}\text{SEQ}}$ can do so by enumerating the missing preconditions through different transitions. The TR contains an example.

We now turn to convergence properties. It is easy to show that partial variable merges can force $h^{\mathcal{C}\text{SEQ}}$ to converge.

Theorem 3. *For every planning task Π with $\mathcal{V}(s_*) = \mathcal{V}$, there exists a set of conjunctions $\mathcal{C}$ s.t. $h^{\mathcal{C}\mathit{SEQ}} = h^*$.*

Proof (sketch). A suitable $\mathcal{C}$ is $\mathcal{C} := \mathcal{S}$. The LP underlying $h^{\mathcal{C}\text{SEQ}}$ then boils down to an LP encoding of shortest paths in the state space graph. □

Corollary 1. *For every planning task Π in TNF, there exists a set of conjunctions $\mathcal{C}$ s.t. $h^{\mathit{SEQ}}[\Pi^{\mathcal{C}}] = h^*$.*

4 Potential Heuristics

Equation 1 depends on the state s considered. So SEQ based heuristics need to solve an LP in every search state. *Potential heuristics* solve an LP only once, in the initial state. The LP solution is used to compute *weights* (potentials) that, when combined in a linear fashion, define an admissible heuristic.

Formally, assume a given set $\mathcal{C}$ of conjunctions. A potential heuristic $h^{\text{Pot}}_{\mathcal{C},w}$ is then defined by a weight function $w : \mathcal{C} \mapsto$

$\mathbb{R}$, with $h^{\text{Pot}}_{\mathcal{C},w}(s) = \sum_{c\in\mathcal{C},c\subseteq s} w(c)$. The computational cost of evaluating such a heuristic on a state s is small. But how to find a suitable w guaranteeing admissibility?

For singleton conjunctions, Pommerening et al. [2015] designed an LP encoding guaranteeing consistency and value ≤ 0 on goal states (*goal-awareness*), which implies admissibility. Pommerening et al. [2017] extended this LP to general conjunctions. For pairs of facts, the size of their LP encoding is still polynomially bounded in the size of Π. For arbitrary conjunctions, however, the LP representation may require an exponential number of variables. In fact, Pommerening et al. [2017] have shown that for conjunctions of size larger than two, the construction of desired potential heuristics is computationally hard. Here we explore this direction further. We show that, for arbitrary $\mathcal{C}$, the $\Pi^{\mathcal{C}}$ compilation can be used to compute consistent and goal-aware potential heuristics. We analyze the convergence properties of this approach.

4.1 Potential Heuristics Over Arbitrary $\mathcal{C}$

We show that Pommerening et al.'s [2015] approach for singleton conjunctions, applied to $\Pi^{\mathcal{C}}$, yields the desired w.

We assume that Π is in TNF. $h^{\text{Pot}}_{\mathcal{C},w}$ is consistent if $\sum_{c\in\mathcal{C},c\subseteq s} w(c) \leq \sum_{c\in\mathcal{C},c\subseteq s[\![a]\!]} w(c) + \mathsf{cost}_a$, or equivalently $\sum_{c\in\mathcal{C},c\subseteq s,c\not\subseteq s[\![a]\!]} w(c) - \sum_{c\in\mathcal{C},c\not\subseteq s,c\subseteq s[\![a]\!]} w(c) \leq \mathsf{cost}_a$. If all conjunctions are singletons, $c = \{p\}$, then because of (TNF1) this inequality is equivalent to

$$\sum_{\{p\}\in\mathcal{C}:a\in Cons(p)} w(p) - \sum_{\{p\}\in\mathcal{C}:a\in Prod(p)} w(p) \leq \mathsf{cost}_a \quad (2)$$

Moreover, by (TNF2) there is only a single goal state. So the weights w ensure goal-awareness if

$$\sum_{\{p\}\in\mathcal{C}:p\in s_*} w(p) \leq 0 \quad (3)$$

These equations define an LP, that we denote Pot[Π], whose variables represent the weights w. Any solution to Pot[Π] yields an admissible potential heuristic.

The objective function in Pot[Π] can be freely chosen. Various possible objectives have been explored [Seipp *et al.*, 2015]. Here, we employ two of these: *(O1) maximizing the heuristic value of an individual state s* (through maximizing the weights of the conjunctions true in s); and *(O2) maximizing the average heuristic value over all states* (through maximizing the sum of all weights normalized by the frequency of the conjunctions). (O1) yields a connection to state equation heuristics. We use (O2) to encode the perfect heuristic as a potential heuristic. (Weight maximization requires an upper bound in the presence of dead-ends; we will discuss this as part of convergence in Section 4.2.)

The LP-based weight computation above requires Π to be in TNF. We use the standard transformation method [Pommerening and Helmert, 2015] to obtain a TNF version $\Pi^{\mathcal{C}}_{\text{TNF}}$ from $\Pi^{\mathcal{C}}$.Pommerening and Helmert [2015] have shown that this TNF transformation does not affect consistent and goal-aware fact potential heuristics, i. e., a fact potential heuristic is consistent and goal-aware for $\Pi^{\mathcal{C}}$ if and only if it is for $\Pi^{\mathcal{C}}_{\text{TNF}}$. In other words, every w computed from Pot[$\Pi^{\mathcal{C}}_{\text{TNF}}$] gives a consistent and goal-aware potential heuristic $h^{\text{Pot}}_{\mathcal{C},w}$ for $\Pi^{\mathcal{C}}$. Lemma 1 leads to the desired result:

Theorem 4. *Let Π be any task, and $\mathcal{C}$ be any set of conjunctions. Let w be any solution to* Pot[$\Pi^{\mathcal{C}}_{\text{TNF}}$]. *Then $h^{\text{Pot}}_{\mathcal{C},w}$ is consistent and goal-aware in Π.*

In other words, we can compute an admissible potential heuristic for any conjunction set $\mathcal{C}$ via a detour to $\Pi^{\mathcal{C}}_{\text{TNF}}$.

Pommerening et al.'s [2017] hardness result is reflected in the worst-case growth of $\Pi^{\mathcal{C}}_{\text{TNF}}$. But for cases where $\Pi^{\mathcal{C}}_{\text{TNF}}$ grows polynomially in $|\mathcal{C}|$, Theorem 4 shows that a desired potential heuristic can be computed in polynomial time. In this sense, Theorem 4 identifies a sufficient criterion for the efficient construction of potential heuristics.

It should be noted that not every admissible potential heuristic over conjunctions $\mathcal{C}$ can be constructed from Pot[$\Pi^{\mathcal{C}}_{\text{TNF}}$]. This is the case because Equation (2) in Pot[$\Pi^{\mathcal{C}}_{\text{TNF}}$] does no longer form a necessary condition for the consistency in Π: Pot[$\Pi^{\mathcal{C}}_{\text{TNF}}$] enforces consistency over occurences a^C where C does not fully specify the action application context, while this context is always completely defined when considering Π's transitions. The TR contains a detailed example.

4.2 Convergence

Does there always exist $\mathcal{C}$ for which $h^{\text{Pot}}_{\mathcal{C},w}$ obtained from Pot[$\Pi^{\mathcal{C}}_{\text{TNF}}$] is perfect? The answer is "yes", under objective (O2) maximizing the average heuristic value. The presence of dead-end states causes complications though.

Obviously, the value $h^*(s) = \infty$ for a dead-end s cannot be produced as part of the solution to an LP. Instead, the weights in the LP may diverge: Pot[Π] is not guaranteed to have a solution optimal for (O2). To see this, consider that no transition path starting from a dead-end ever reaches the goal; so, for conjunctions c true only in dead-ends, the weight can be made arbitrarily high while still satisfying consistency. Intuitively, the LP encoding imposes constraints on solution paths over conjunctions, and diverges where such a path does not exist. For that reason, Seipp et al. [2015] introduce a modified LP with the additional constraints $w(c) \leq U$, where $U \in \mathbb{R}$ is a parameter. We denote that LP by Pot[Π, U].

Intuitively, U is a cut-off value on the cost of solutions considered in the LP. Convergence is achieved below U:

Theorem 5. *Let Π be any task in TNF, and $U \in \mathbb{R}^+_0$. Then there exists a set $\mathcal{C}$ of conjunctions s.t., with w being a solution to* Pot[$\Pi^{\mathcal{C}}_{\text{TNF}}, U$] *optimal for (O2), $h^{\text{Pot}}_{\mathcal{C},w}(s) = h^*(s)$ for all states s with $h^*(s) \leq U$.*

Proof (sketch). A set of conjunctions satisfying the claim is again $\mathcal{C} := \mathcal{S}$, the set of all states in the task. Pot[$\Pi^{\mathcal{C}}_{\text{TNF}}, U$] then boils down to an LP encoding of paths in the state space of Π, with Equation 2 bounding the value of a state by its successor states. Objective (O2) makes sure that, up to U, the exact shortest path length is returned. □

A simple trick now suffices to obtain h^* globally. We pessimistically interpret the cut-off U as a dead-end indicator, defining $h^{\text{Pot}}_{\mathcal{C},w,U}(s) := h^{\text{Pot}}_{\mathcal{C},w}(s)$ if $h^{\text{Pot}}_{\mathcal{C},w}(s) < U$ and $h^{\text{Pot}}_{\mathcal{C},w,U}(s) := \infty$ otherwise. We then need to choose a cut-off that will never apply on solvable states, $U > h^*(s)$ for all s with $h^*(s) < \infty$. This is the case for $U^* := \left(\prod_{v\in\mathcal{V}} |\mathcal{D}_v| \cdot \max_{a\in\mathcal{A}} \mathsf{cost}_a\right) + 1$.

Corollary 2. *Let Π be any task in TNF. Then there exists a set $\mathcal{C}$ of conjunctions s.t., with w being a solution to* $\text{Pot}[\Pi^{\mathcal{C}}_{\text{TNF}}, U^*]$ *optimal for (O2), $h^{\text{Pot}}_{\mathcal{C},w,U^*} = h^*$.*

For the simpler purpose of detecting all dead-end states, it is not necessary to use the exceedingly large constant U^*. Following previous work on potential heuristics for dead-end detection [Seipp *et al.*, 2016b], we instead consider the task Π_0 identical to Π except that all actions are assigned cost 0. Clearly, $h^*[\Pi_0](s) = \infty$ iff $h^*(s) = \infty$, i. e., $h^*[\Pi_0]$ detects all dead-ends in Π. But all solvable states s have $h^*[\Pi_0](s) = 0$, so setting U^* to any number > 0 results in $h^{\text{Pot}}_{\mathcal{C},w,U^*}$ that converges to $h^*[\Pi_0]$ as per Corollary 2. We will denote potential heuristics constructed this way as $u^{\text{Pot}}_{\mathcal{C},w}$.

4.3 Relation to the State Equation

Pommerening et al. [2015] have shown that $\text{Pot}[\Pi]$ under objective (O1) for a state s is the dual of the state equation LP for s. By the strong duality theorem for linear programs, the two heuristics therefore have identical values on s.

Beyond individual states, the heuristics differ though: on states other than s, the potential heuristic merely gives a lower bound on $h^{\text{SEQ}}(s)$. In fact, there exist tasks and conjunction sets where *no* potential heuristic $h^{\text{Pot}}_{\mathcal{C},w}$ equals $h^{\text{SEQ}}[\Pi^{\mathcal{C}}]$ on all states. The TR specifies such an example.

5 Refining the State Equation

We have shown that LP heuristics converge to h^* for suitable conjunctions $\mathcal{C}$. As an application of this property, for the rest of the paper we consider proving unsolvability, through *nogood learning* (dead-end detection) using LP heuristics. This is motivated by the success of LP heuristics in the Unsolvability-IPC [Seipp *et al.*, 2016b]. Convergence is essential as the heuristic must be able to represent arbitrary sets of dead-end states in the limit.

The key step in nogood learning with a heuristic h is *refinement*: given a dead-end state s not pruned by h so far, refine h into h' that prunes s. Whether this can be done, and how to best do it in practice, depends crucially on which heuristic h is used. Steinmetz and Hoffmann [2017] have shown how to select new conjunctions for critical-path heuristics. Here we introduce a new refinement method selecting conjunctions suitable to refine the state equation.

To provide an overview, we next describe the forms of nogood learning we use in our experiments. Then we introduce our refinement method.

5.1 Nogood Learning

We start with $\mathcal{C}$ containing the singleton conjunctions. We experiment with three different forms of nogood learning:

$\mathcal{C}_{\mathcal{I}}$Seq : Proves the task unsolvable on the initial state. We iteratively apply refinement, adding new conjunctions into $\mathcal{C}$, until $h^{\text{SEQ}}[\Pi^{\mathcal{C}}](s_{\mathcal{I}}) = \infty$.

$\mathcal{C}_{\mathbf{S}}$Seq : Forward-search nogood learning as per Steinmetz and Hoffmann [2017], using $h^{\text{SEQ}}[\Pi^{\mathcal{C}}]$. A depth-oriented search calls refinement when backtracking out of a state s then known to be a (undetected) dead-end. The refined $h^{\text{SEQ}}[\Pi^{\mathcal{C}}]$ may generalize to dead-ends not yet encountered, reducing the future search space.

$\mathcal{C}_{\mathbf{S}}$Pot : Similar to $\mathcal{C}_{\mathbf{S}}$Seq, but using potential heuristics. We maintain a collection of such heuristics. The refinement steps work as before, finding a larger set $\mathcal{C}$ suitable for $h^{\text{SEQ}}[\Pi^{\mathcal{C}}]$; but now we also add a new potential heuristic $u^{\text{Pot}}_{\mathcal{C},w}$ (optimal under (O1) for s considered in the refinement) into the collection. To check whether a new state is a dead-end, only the potential heuristics are evaluated, which does not require any LP solving.

5.2 Refinement Method

We assume Π to be in TNF. Let $\mathcal{C}$ be any set of conjunctions, and s any dead-end where $h^{\text{SEQ}}[\Pi^{\mathcal{C}}](s) \neq \infty$. We need to extend $\mathcal{C}$ to $\mathcal{C}' \supseteq \mathcal{C}$ such that $h^{\text{SEQ}}[\Pi^{\mathcal{C}'}](s) = \infty$. We do so by iteratively finding a conjunction $x \notin \mathcal{C}$ whose SEQ constraint is not satisfied by a current LP solution. We set $\mathcal{C} := \mathcal{C} \cup \{x\}$. If $h^{\text{SEQ}}[\Pi^{\mathcal{C}}](s) = \infty$, we stop; else, we iterate. By Corollary 1, the termination condition must hold eventually.

It remains to show how to choose x. Denote by $\text{Seq}[\Pi^{\mathcal{C}}]$ the LP underlying $h^{\text{SEQ}}[\Pi^{\mathcal{C}}](s)$. The refinement is based on a concrete solution Count to $\text{Seq}[\Pi^{\mathcal{C}}]$. Consider (1) the $\Pi^{\mathcal{C}}$ actions *selected* by Count, i. e., $\{a^C \in \mathcal{A}^{\mathcal{C}} \mid \mathsf{Count}_{a^C} > 0\}$. Additionally consider (2) two auxiliary actions a_s and a_*, representing in $\Pi^{\mathcal{C}}$ the current state and the goal, i. e., $pre_{a_s} = \emptyset$, $\mathit{eff}_{a_s} = s^{\mathcal{C}}$, $pre_{a_*} = s^{\mathcal{C}}_*$, and $\mathit{eff}_{a_*} = \emptyset$. We denote by $\mathcal{A}^{\mathcal{C}}_{\mathsf{Count}}$ the actions of (1) and (2). Our key observation is that we can find an action $a_0^{C_0} \in \mathcal{A}^{\mathcal{C}}_{\mathsf{Count}}$ whose precondition is not supported by $\mathcal{A}^{\mathcal{C}}_{\mathsf{Count}}$: *(*) for all* $a^C \in \mathcal{A}^{\mathcal{C}}_{\mathsf{Count}}$, $\mathsf{regr}(pre_{a_0^{C_0}}, a^C) \not\subseteq pre_{a^C}$. (Recall that the actions in $\Pi^{\mathcal{C}}$ represent action occurrences in the original task, carrying the context of application.)

We show (*) as follows. Consider the graph with nodes $\mathcal{A}^{\mathcal{C}}_{\mathsf{Count}}$ and edges $a^C \to a_0^{C_0}$ for every $a_0^{C_0}$ and a^C where $\mathsf{regr}(pre_{a_0^{C_0}}, a^C) \subseteq pre_{a^C}$. In this graph, every path from a_s to a_* would correspond to a plan for s. Since s is a dead-end, such path cannot exist. Hence there exists at least one node which is not connected to a_s. If (*) is not satisfied for any $a_0^{C_0}$, then every node must have an incoming edge. But then, those actions in $\mathcal{A}^{\mathcal{C}}_{\mathsf{Count}}$ that are disconnected from a_s must form at least one cycle, i. e., omitting the "C" superscripts for readability, there must be $a_1, \dots, a_n \in \mathcal{A}^{\mathcal{C}}_{\mathsf{Count}} \setminus \{a_s\}$ s.t. $\mathsf{regr}(pre_{a_{i+1}}, a_i) \subseteq pre_{a_i}$ and $\mathsf{regr}(pre_{a_1}, a_n) \subseteq pre_{a_n}$. As Π is in TNF, every fact produced along $a_1, \dots, a_n$ is also consumed, and vice versa. Thus we can obtain another feasible solution Count' to the LP such that $a_i \notin \mathcal{A}^{\mathcal{C}}_{\mathsf{Count}'}$ for at least one i, by reducing the values Count_{a_i} appropriately. Repeatedly applying this step will eventually remove all cycles, leaving us with the desired action $a_0^{C_0}$.

Let P now denote the precondition of $a_0^{C_0}$, projected onto the original variables $\mathcal{V}$. Consider $\Pi^{\mathcal{C}'}$ for $\mathcal{C}' = \mathcal{C} \cup \{P\}$, and the π_P-variable corresponding to P. Since Π is in TNF, $\langle \pi_P, 1 \rangle$ is consumed by $a_0^{C_0}$. However, by (*) Count does not include any action that produces $\langle \pi_P, 1 \rangle$. In other words, Count violates the constraint corresponding to $\langle \pi_P, 1 \rangle$ in $\text{Seq}[\Pi^{\mathcal{C}'}]$. Hence $P \notin \mathcal{C}$ and we can set $x := P$.

We employ two optimizations. 1) we minimize P, starting with $x = P$ and greedily removing facts p from x so long as

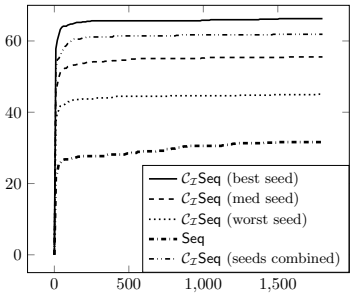

Figure 2: Coverage (in %) over time (in s) for different variable orders in conjunction generation, see text.

the necessary properties are preserved. 2) we consider not a single $a_0^{C_0}$, but all actions with that profile, and add a conjunction x for each. This results in fewer refinement iterations.

6 Experiments

Our implementation is in Fast Downward (FD) [Helmert, 2006]. We use the UIPC'16 benchmarks, as well as unsolvable resource-constrained (RCP) benchmarks [Nakhost *et al.*, 2012; Steinmetz and Hoffmann, 2017]. All experiments were run on machines equipped with Intel Xeon E5-2660 CPUs, with runtime (memory) limits of 30 minutes (4 GB).

Similar to earlier works on the $\Pi^\mathcal{C}$-compilation [Keyder *et al.*, 2014], we cope with the worst-case explosion by imposing a size limit M on the ratio $|\mathcal{A}^\mathcal{C}|/|\mathcal{A}|$. Once $\Pi^\mathcal{C}$ reaches the limit M, we disable the generation of new conjunctions. We experimented with $M \in \{2, 4, 8, \ldots, 1024, \infty\}$, where for $M = \infty$ the size of $\Pi^\mathcal{C}$ is not limited.

Figure 2 sheds light on an implementation detail that turns out to be important. Optimization 1) described in the previous section leaves open the order in which to remove facts p from x. We make this choice by fixing a variable order a priori. That order has a large impact on performance. Figure 2 compares the results for $\mathcal{C}_\mathcal{I}$Seq and $x = \infty$, for five randomly generated orders, picking the per-instance best, median, and worst variable order. The variance in coverage is large.

To counteract this brittleness, all our configurations in what follows *combine* the five variable orders, maintaining for each a separate conjunction set. Refinement works on all these sets, interleaving the individual refinement steps and stopping as soon as any of them succeeds. As can be seen in Figure 2 ("seeds combined"), this performs almost as well as the hypothetical per-instance best configuration. Considering fewer orders negatively affects coverage. Coverage remains stable for up to 10 orders, but starts to drop off eventually due to the additional overhead introduced with every order.

Table 1 shows our main coverage results, comparing our techniques to baselines and the state of the art. Here, $h^\mathcal{C}$ is the forward-search nogood learning algorithm of Steinmetz and Hoffmann [2017]; PDB is a component of Aidos [Seipp *et al.*, 2016b], the winner of UIPC'16, evaluated separately to consider algorithms rather than systems.

Consider first the comparison of our algorithms to the baselines, Seq and Pot in Table 1, that use the same heuristics but over the singleton conjunctions only, without any refinement. On the UIPC benchmarks, Seq and Pot dominate in the overall, but are outperformed by our techniques in Document-Transfer, NoMystery, and Rovers. On the RCP benchmarks, our techniques are vastly better. These observations hold regardless of our configuration, with the single exception of $\mathcal{C}_\mathcal{I}$Seq in UIPC Rovers. We remark that the bad performance of our methods in the BagGripper and BagTransport domains is only due to the overhead of maintaining five different conjunction sets (cf. above); when maintaining a single set $\mathcal{C}$, we get the same coverage here as Seq respectively Pot.

Domain #		Blind	h^1	h^C	PDB	Seq	Pot	$\mathcal{C}_S$Seq 128	$\mathcal{C}_S$Seq 256	$\mathcal{C}_S$Seq ∞	$\mathcal{C}_S$Pot 128	$\mathcal{C}_S$Pot 256	$\mathcal{C}_S$Pot ∞	Seq $\mathcal{I}$	Seq $\mathcal{C}_\mathcal{I}$
Unsolvability-IPC (UIPC) 2016 Benchmarks															
BagBar	20	**12**	8	0	**12**	4	**12**	0	0	0	0	0	0	0	0
BagGri	25	4	3	2	3	**14**	8	2	2	2	2	2	2	**14**	2
BagTra	29	7	6	6	7	**22**	**22**	19	19	19	19	19	19	**22**	19
Bottle	25	10	21	9	19	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**
CaveDi	25	7	7	**8**	7	**8**	**8**	7	7	2	4	3	2	1	5
Chess	23	5	5	2	5	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**
Diagno	20	4	5	**9**	5	4	4	4	4	2	3	3	2	0	2
DocTra	20	5	7	5	**12**	6	5	9	9	7	9	9	7	0	7
NoMys	20	2	2	11	11	1	2	**12**	**12**	**12**	11	11	11	0	11
Rovers	20	7	7	**12**	**12**	6	7	10	8	8	10	10	10	0	4
TPP	30	17	16	19	**24**	11	17	17	17	17	17	16	16	2	16
PegSol	15	5	5	4	5	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**
PegSol	24	**24**	**24**	14	**24**	**24**	**24**	20	20	4	16	14	4	0	4
SlidTil	20	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	0	0
Tetris	20	10	5	5	10	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
$\sum$	336	129	131	116	166	193	**202**	193	191	166	184	180	166	122	153
Unsolvable Resource-Constrained Planning (RCP) Benchmarks															
NoMys	150	27	53	130	**149**	15	27	137	131	131	140	132	131	0	137
Rovers	150	3	7	**142**	93	1	3	117	118	117	120	121	121	0	110
TPP	25	6	5	13	**20**	0	5	9	9	9	8	8	8	0	9
$\sum$	325	36	65	**285**	262	16	35	263	258	257	268	261	260	0	256
$\sum\sum$	661	165	196	401	428	209	237	**456**	449	423	452	441	426	122	409

Table 1: Coverage. Best results in **bold**. h^1: search with h^1 heuristic (for dead-end detection). h^C: nogood learning as per Steinmetz and Hoffmann (see text). PDB: dead-end PDB of Aidos, as per Seipp et al. (see text). Seq and Pot: search with state equation heuristic, respectively potential heuristic, over singleton conjunctions. Seq, $\mathcal{I}$: h^{SEQ} on initial state only, w/o learning; $\mathcal{C}_\mathcal{I}$ same but w/ learning.

Considering h^{SEQ} on the initial state only, without vs. with learning (the rightmost two columns), shows that the learned larger conjunctions yield a dramatic increase in unsolvability-detection power, despite the quick-or-not-at-all performance profile observed in Figure 2. Indeed, the number of conjunctions needed to prove $s_\mathcal{I}$ unsolvable here is typically small. The maximal ratio $|\mathcal{C}|/\sum_{v\in\mathcal{V}}|\mathcal{D}_v|$ required is 1.66.

Comparing to the state of the art, UIPC NoMystery is the only domain where the coverage of (the best of) our new methods is strictly higher (by the smallest margin, +1) than that of any competitor. The main advantage of our methods is that they combine both, the strength of LP heuristics on the UIPC benchmarks, and that of conjunction-learning on RCP benchmarks: they are the only configurations with near-top performance in both benchmark categories. The "$\sum\sum$" row of Table 1 illustrates this (but should be taken with a grain of salt given the different numbers of instances per domain).

Comparing our configurations against each other, forward-search nogood learning consistently outperforms proving unsolvability on the initial state. The large limits $M = 256$ and $M = \infty$ are almost consistently worse than $M = 128$. Somewhat surprisingly, potential heuristics hardly ever improve over the state equation. Figure 3 elucidates the latter.

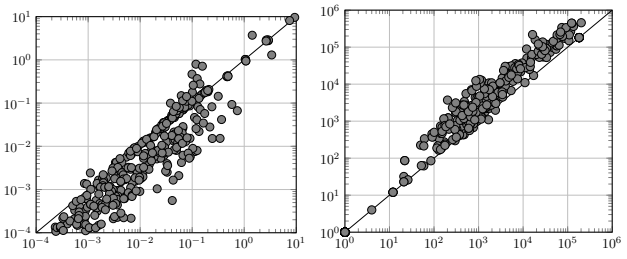

Figure 3: Per-state runtime (left) and number of visited states (right), of $\mathcal{C}_{\mathrm{S}}\mathsf{Seq}$ (x-axes) vs. $\mathcal{C}_{\mathrm{S}}\mathsf{Pot}$ (y-axes).

$\mathcal{C}_{\mathrm{S}}\mathsf{Seq}$ has the edge in search space size, due to its higher pruning power; while potential heuristics are faster. Yet the former effect tends to be larger than the latter one.

7 Conclusion

LP heuristics yield powerful approximations in planning. We contributed insights on their definition over conjunctions, pertaining to the natural approach of using the $\Pi^{\mathcal{C}}$ compilation, its relation to previous techniques, convergence, and nogood learning via conjunction refinement.

Interesting avenues for future work are, e. g., conjunction refinement for optimal planning during A* search, and conjunction refinement for satisficing planning targeted at Seipp et al.'s [2016a] descending and dead-end avoiding heuristics.

Acknowledgments

We thank the anonymous reviewers, whose comments helped to improve the paper. This work was partially supported by the German Research Foundation (DFG), under grant HO 2169/5-1, as well as by the German Federal Ministry of Education and Research (BMBF) through funding for CISPA, under grant no. 16KIS0656.

References

[Bäckström and Nebel, 1995] Christer Bäckström and Bernhard Nebel. Complexity results for SAS$^+$ planning. *Computational Intelligence*, 11(4):625–655, 1995.

[Bonet and van den Briel, 2014] Blai Bonet and Menkes van den Briel. Flow-based heuristics for optimal planning: Landmarks and merges. In *Proc. ICAPS'14*, pages 47–55, 2014.

[Bonet, 2013] Blai Bonet. An admissible heuristic for SAS+ planning obtained from the state equation. In *Proc. IJCAI'13*, pages 2268–2274, 2013.

[Haslum, 2009] Patrik Haslum. $h^m(P) = h^1(P^m)$: Alternative characterisations of the generalisation from $h^{\max}$ to h^m. In *Proc. ICAPS'09*, pages 354–357, 2009.

[Haslum, 2012] Patrik Haslum. Incremental lower bounds for additive cost planning problems. In *Proc. ICAPS'12*, pages 74–82, 2012.

[Helmert, 2006] Malte Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[Helmert, 2009] Malte Helmert. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, 173:503–535, 2009.

[Keyder *et al.*, 2014] Emil Keyder, Jörg Hoffmann, and Patrik Haslum. Improving delete relaxation heuristics through explicitly represented conjunctions. *Journal of Artificial Intelligence Research*, 50:487–533, 2014.

[Nakhost *et al.*, 2012] Hootan Nakhost, Jörg Hoffmann, and Martin Müller. Resource-constrained planning: A Monte Carlo random walk approach. In *Proc. ICAPS'12*, pages 181–189, 2012.

[Pommerening and Helmert, 2015] Florian Pommerening and Malte Helmert. A normal form for classical planning tasks. In *Proc. ICAPS'15*, pages 188–192, 2015.

[Pommerening *et al.*, 2014] Florian Pommerening, Gabriele Röger, Malte Helmert, and Blai Bonet. LP-based heuristics for cost-optimal planning. In *Proc. ICAPS'14*, pages 226–234, 2014.

[Pommerening *et al.*, 2015] Florian Pommerening, Malte Helmert, Gabriele Röger, and Jendrik Seipp. From non-negative to general operator cost partitioning. In *Proc. AAAI'15*, pages 3335–3341, 2015.

[Pommerening *et al.*, 2017] Florian Pommerening, Malte Helmert, and Blai Bonet. Higher-dimensional potential heuristics for optimal classical planning. In *Proc. AAAI'17*, pages 3636–3643, 2017.

[Seipp *et al.*, 2015] Jendrik Seipp, Florian Pommerening, and Malte Helmert. New optimization functions for potential heuristics. In *Proc. ICAPS'15*, pages 193–201, 2015.

[Seipp *et al.*, 2016a] Jendrik Seipp, Florian Pommerening, Gabriele Röger, and Malte Helmert. Correlation complexity of classical planning domains. In *Proc. IJCAI'16*, pages 3242–3250, 2016.

[Seipp *et al.*, 2016b] Jendrik Seipp, Florian Pommerening, Silvan Sievers, and Martin Wehrle. Fast Downward Aidos. In *UIPC 2016 planner abstracts*, pages 28–38, 2016.

[Steinmetz and Hoffmann, 2017] Marcel Steinmetz and Jörg Hoffmann. State space search nogood learning: Online refinement of critical-path dead-end detectors in planning. *Artificial Intelligence*, 245:1 – 37, 2017.

[Steinmetz and Hoffmann, 2018] Marcel Steinmetz and Jörg Hoffmann. LP heuristics over conjunctions: Compilation, convergence, nogood learning (technical report). Technical report, Saarland University, 2018. Available at `http://fai.cs.uni-saarland.de/hoffmann/papers/ijcai18b-tr.pdf`.

[van den Briel *et al.*, 2007a] Menkes van den Briel, J. Benton, Subbarao Kambhampati, and Thomas Vossen. An LP-based heuristic for optimal planning. In *Proc. CP'07*, pages 651–665, 2007.

[van den Briel *et al.*, 2007b] Menkes van den Briel, Subbarao Kambhampati, and Thomas Vossen. Fluent merging: A general technique to improve reachability heuristics and factored planning. In *Proc. HDIP'07*, pages 20–24, 2007.