LP Heuristics over Conjunctions: Compilation, Convergence, Nogood Learning (Technical Report)

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; Haslum, 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.

2 Preliminaries

We consider planning tasks in FDR notation [Bäckström and Nebel, 1995; Helmert, 2009]. A *task* is a tuple $\Pi = \langle \mathcal{V}, \mathcal{A}, s_{\mathcal{I}}, s_* \rangle$. $\mathcal{V}$ is a set of *variables* v, each associated with a finite domain $\mathcal{D}_v$. A *state* is a complete assignment to V.

[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.

$s_{\mathcal{I}}$ 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 $\pi = \langle a_1, \dots, a_n \rangle$ 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, $P^{\mathcal{C}} := P \cup \{\langle \pi_c, 1 \rangle \mid c \in \mathcal{C}, c \subseteq P\} \cup \{\langle \pi_c, 0 \rangle \mid c \in \mathcal{C}, c \not\| P\}$ 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) $\mathit{eff}_{a^C} = \mathit{eff}_a \cup \{\langle \pi_c, 1 \rangle \mid c \in C\} \cup \{\langle \pi_{c'}, 0 \rangle \mid c' \in \mathcal{C}, c' || pre_{a^C}, c' \not\| \mathit{eff}_a\}$, and (iii) $\mathsf{cost}_{a^C} = \mathsf{cost}_a$.*

[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.

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.

Following previous works on the $\Pi^{\mathcal{C}}$-compilation, we interpret heuristics for $\Pi^{\mathcal{C}}$, written $h[\Pi^{\mathcal{C}}]$, as heuristics for Π by mapping every state s of Π to the state in $\Pi^{\mathcal{C}}$ obtained by augmenting s with the π_c-variable assignments according to whether or not c is satisfied in s, i. e., we define $h[\Pi^{\mathcal{C}}](s) := h[\Pi^{\mathcal{C}}](s^{\mathcal{C}})$.

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. Let h be any consistent heuristic for Π. Let t be any state of $\Pi^{\mathcal{C}}$. We denote by $t|_{\mathcal{V}}$ the projection of t onto the variables $\mathcal{V}$. Let a^C be any action occurrence in $\Pi^{\mathcal{C}}$ that is applicable in t. Obviously, a is applicable in $t|_{\mathcal{V}}$ and it holds that $t|_{\mathcal{V}}[\![a]\!] = (t[\![a^C]\!])|_{\mathcal{V}}$. Therefore, $h(t|_{\mathcal{V}}) \leq h(t|_{\mathcal{V}}[\![a]\!]) + \mathsf{cost}_a$ implies $h(t) \leq h(t[\![a^C]\!]) + \mathsf{cost}_{a^C}$, which shows that h is indeed consistent in $\Pi^{\mathcal{C}}$.

Let h be any consistent heuristic for $\Pi^{\mathcal{C}}$. Let s be any state of Π, $a \in \mathcal{A}$ be s.t. $pre_a \subseteq s$. Define $C := \{c \in \mathcal{C} \mid c \subseteq s[\![a]\!], c \cap \mathit{eff}_a \neq \emptyset\}$. Obviously, C is compatible; for every $c \in C$, it holds that $\mathsf{regr}(c, a) \neq \bot$; and for every $c' \in (\mathcal{C} \setminus C)$, either $\mathsf{regr}(c', a) = \bot$ or $\mathsf{regr}(c', a) \not\subseteq s$ and thus $\mathsf{regr}(c', a) \not\subseteq (pre_a \cup \bigcup_{c \in C} \mathsf{regr}(c, a))$. Hence, $a^C \in \mathcal{A}^{\mathcal{C}}$. Moreover, the selection of C ensures that a^C is applicable in $s^{\mathcal{C}}$ and that for every $c \subseteq s[\![a]\!]$, $s^{\mathcal{C}}[\![a^C]\!](\pi_c) = 1$. For every $c \not\subseteq s[\![a]\!]$, it holds that $c \notin C$ and either $s^{\mathcal{C}}(\pi_c) = 0$, or $c \subseteq s$ and $c \not\| \mathit{eff}_a$, i. e., $c || pre_{a^C}$ and $\mathit{eff}_{a^C}(\pi_c) = 0$. In both cases, $s^{\mathcal{C}}[\![a^C]\!](\pi_c) = 0$ follows. We get $h(s) = h(s^{\mathcal{C}}) \leq h(s^{\mathcal{C}}[\![a^C]\!]) + \mathsf{cost}_{a^C} = h((s[\![a]\!])^{\mathcal{C}}) + \mathsf{cost}_{a^C} = h(s[\![a]\!]) + \mathsf{cost}_a$, what shows the claim. □

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 $Prod(p)$ ($Cons(p)$) the set of all actions that produce (consume) p. Then the SEQ for p is

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

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 $h^{\text{SEQ}}(s)$ 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 $h^{\text{SEQ}}(s)$ gives the respective 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.*

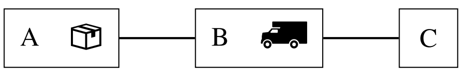

Figure 1: Initial state in the transportation example.

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* $\mathit{unloaded}$ *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,* $h^{SEQ}[\Pi^\mathcal{C}](s_\mathcal{I}) = 5$*.*

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 the sake of readability, within this section, we only provide the principle ideas underlying the proofs of our results. The detailed proofs are available in Appendix A.

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 state-changing abstract transition $x \xrightarrow{a} x'$ induced by a, i. e.,

$$\sum_{a \in PProd(c)} \mathsf{Count}_a^{x \to c} - \sum_{a \in PCons(c)} \mathsf{Count}_a^{c \to x} \geq \Delta_c(s) \quad (2)$$

$\Delta_c(s)$ is defined similarly to single facts: $\Delta_c(s) = 1$ if $c \not\subseteq s$ and $c \subseteq s_*$; $\Delta_c(s) = -1$ if $c \subseteq s$ and $c \not\subseteq s_*$; and $\Delta_c(s) = 0$ otherwise. The sets $PProd(c)$ and $PCons(c)$ thereby identify exactly the labels of the state-changing transitions for conjunction c. Finally, these separate counters are related back to the main action counters via the following link constraint:

$$\sum \mathsf{Count}_a^{x \to x'} \leq \mathsf{Count}_a \quad (3)$$

where the sum is over all state-changing abstract transitions induced by a in the considered variable merge. For every partial variable merge with corresponding variable set V and every action a where the effect of a on any conjunction over the variables V is independent of the application context, i. e., for every V and a such that $V \subseteq \mathcal{V}(pre_a)$ and there exists a conjunction $c \in \mathcal{C}$ with $\mathcal{V}(c) = V$ and either (a) $c \subseteq pre_a$ and $c \not\parallel \mathit{eff}_a$, or (b) $pre_a \subseteq \mathsf{regr}(c, a)$, a can only label a single transition in the respective partial variable merge. Moreover, every application of a necessarily implies the execution of this transition. Hence, one can replace the inequality in Equation (3) by an equality, strengthening the overall LP. In the actual implementation, and as done in Bonet and van den Briel's [2014] original proposal, the respective transition-count variable can directly and equivalently be replaced by Count_a.

We denote by $h^{\mathcal{C}\text{SEQ}}$ the heuristic extending Equation (1) for facts by Equation (2) and Equation (3) for conjunctions.

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

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* $h^{\mathcal{C}'\text{SEQ}}(s) \geq h^{SEQ}[\Pi^{\mathcal{C}}](s)$ *for all states* s*,* $\mathcal{C}'$ *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.

Example 3. *Consider the task with variables* v_1 *and* v_2 *and domains* $\mathcal{D}_{v_1} = \mathcal{D}_{v_2} = \{0, 1, 2\}$*; actions:* a_1 *requires that* $v_2 = 1$ *and sets* $v_1 = 1$*,* a_2 *requires that* $v_1 = 1$ *and sets* $v_2 = 1$*, and* a_3 *with empty precondition and effect* $v_1 = 2$*; initial state* $s_{\mathcal{I}} = \{\langle v_1, 0\rangle, \langle v_2, 0\rangle\}$ *and goal* $s_* = \{\langle v_1, 1\rangle, \langle v_2, 1\rangle\}$*. Assume that* $\mathcal{C}$ *contains all conjunctions.*

Since the abstract states corresponding to the conjunctions $s_{\mathcal{I}}$ *and* s_* *are not connected, it is not possible to satisfy Equation* (2) *for* s_* *without violating the state equation constraint of any other conjunction. Hence,* $h^{\mathcal{C}SEQ}(s_{\mathcal{I}}) = \infty$.

However, $h^{SEQ}[\Pi^{\mathcal{C}}](s_{\mathcal{I}}) < \infty$*: let* $\mathsf{Count}_{a_1^{\{s_*\}}} = 1$, $\mathsf{Count}_{a_2^{\{s_*\}}} = 1$, $\mathsf{Count}_{a_3^{\emptyset}} = 1$. *It is straightforward to verify that* Count *satisfies Equation* (1) *for all facts* $\langle v, d\rangle$ *and* π_c *value-assignments. The action occurrence* $a_3^{\emptyset}$ *is required to satisfy the constraint for* $\langle \pi_{s_{\mathcal{I}}}, 0\rangle$.

Combining the task from Example 3 with the one from the proof of Theorem 2 shows the existence of non-TNF Π and $\mathcal{C}$ for which $h^{\text{SEQ}}[\Pi^{\mathcal{C}}]$ and $h^{\mathcal{C}\text{SEQ}}$ are incomparable.

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.* $h^{\mathcal{C}SEQ} = h^*$.

Proof. A suitable $\mathcal{C}$ is $\mathcal{C} := \mathcal{S}$, i. e., the set of all states in the given task Π. The LP underlying $h^{\mathcal{C}\text{SEQ}}$ then boils down to an LP encoding of shortest paths in the state space graph, essentially a special case of the min-cost flow problem. The variables encode state-transition weights. The constraints ensure that the difference between the incoming and outgoing flow at every state is ≥ 0, respectively ≥ -1 (≥ 1) at initial (goal) state. Minimizing action (and thus state-transition) weights, equality holds in optimal solutions. The condition $\mathcal{V}(s_*) = \mathcal{V}$ is required because $\Delta_c(s) = 1$ may only hold if $c \subseteq s_*$. Thus, in order for the state space graph, encoded as the variable merge over variables $\mathcal{V}$, to actually contain a *sink* state, it must hold $\mathcal{V}(s_*) = \mathcal{V}$. □

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

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 comparatively 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

$$\sum_{\{p\}\in\mathcal{C}:a\in Cons(p)} w(p) - \sum_{\{p\}\in\mathcal{C}:a\in Prod(p)} w(p) \leq \mathsf{cost}_a \quad (4)$$

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

$$\sum_{\{p\}\in\mathcal{C}:p\in s_*} w(p) \leq 0 \quad (5)$$

These equations define an LP, that we denote $\text{Pot}[\Pi]$, whose variables represent the weights w. Any solution to $\text{Pot}[\Pi]$ yields an admissible potential heuristic.

The objective function in $\text{Pot}[\Pi]$ 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: $\max \sum_{c \in \mathcal{C}, c \subseteq s} w(c)$

(O2) *Maximizing the average heuristic value over all states.* This requires to normalize the weight associated with a conjunction c with the *frequency* of c, i. e., by the fraction of states in which c is satisfied. The frequency of c is syntactically defined by $freq(c) = 1/\Pi_{\langle v,d\rangle \in c}|\mathcal{D}_v|$. (O2) then reads $\max \sum_{c \in \mathcal{C}} freq(c) \cdot w(c)$.

(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 $\Pi^{\mathcal{C}}$: add $\top$ to the domain of every $v \in \mathcal{V}^{\mathcal{C}}$; for every $v \in \mathcal{V}^{C}$ and value d, create a 0-cost action $unset_{\langle v,d\rangle}$ with precondition $\{v = d\}$ and effect $\{v = \top\}$; for every $a^C \in \mathcal{A}^C$ and $v \in (\mathcal{V}(\mathit{eff}_{a^C}) \setminus \mathcal{V}(pre_{a^C}))$, add $\langle v = \top\rangle$ to the precondition; add $\langle v, \top\rangle$ to the goal for all $v \in (\mathcal{V}^{\mathcal{C}} \setminus \mathcal{V}(s^{\mathcal{C}}_*))$. 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, for every set $\hat{\mathcal{C}}$ over singleton conjunctions over $\Pi^{\mathcal{C}}_{\text{TNF}}$'s facts, and for every $\hat{w} : \hat{\mathcal{C}} \mapsto \mathbb{R}$ satisfying Pot$[\Pi^{\mathcal{C}}_{\text{TNF}}]$, it follows that $h^{\text{Pot}}_{\hat{\mathcal{C}},\hat{w}}$ is a consistent and goal-aware potential heuristic in $\Pi^{\mathcal{C}}$. For an arbitrary set of conjunctions $\mathcal{C}$, it is therefore straightforward to construct a potential heuristic over $\mathcal{C}$ for Π, via a detour to $\Pi^{\mathcal{C}}_{\text{TNF}}$:

Theorem 4. *Let* Π *be any task, and* $\mathcal{C}$ *be any set of conjunctions containing at least all singleton conjunctions. Let* $\hat{\mathcal{C}}$ *be the set of singleton conjunctions over* $\Pi^{\mathcal{C}}_{\text{TNF}}$*'s facts, and* $\hat{w} : \hat{\mathcal{C}} \mapsto \mathbb{R}$ *be any solution to* Pot$[\Pi^{\mathcal{C}}_{\text{TNF}}]$*. There exists* $w : \mathcal{C} \mapsto \mathbb{R}$ *such that* $h^{\text{Pot}}_{\mathcal{C},w}(s) = h^{\text{Pot}}_{\hat{\mathcal{C}},\hat{w}}(s^{\mathcal{C}})$ *for all states* $s \in \mathcal{S}$.

Proof. In the following, we will assume that $\hat{\mathcal{C}}$ does not contain $\{\langle \pi_c, d\rangle\}$ for any singleton conjunction $c = \{\langle v, d\rangle\}$. Since $v \in \mathcal{V}^{\mathcal{C}}$, those can be handled directly and equivalently in the $\Pi^{\mathcal{C}}$-compilation, without having to introduce an auxiliary π_c variable.

We define w as follows. Let $\mathcal{C}_1 \subseteq \mathcal{C}$ denote the singleton conjunctions in consideration. For $c \in \mathcal{C}_1$, we define $w(c) := \hat{w}(c)$, with one exception, see below. Handling conjunctions is a bit more difficult: the facts $\langle \pi_c, 0\rangle$ allow $\hat{w}$ to also associate weights with the negation of conjunctions; but this is not possible with w. Consider the sum over the weights of all negated conjunctions: $\delta = \sum_{c \in (\mathcal{C} \setminus \mathcal{C}_1)} \hat{w}(\{\langle \pi_c, 0\rangle\})$. Let $v_\delta \in \mathcal{V}$ be an arbitrary variable. We set $w(\{\langle v_\delta, d\rangle\}) := \hat{w}(\{\langle v_\delta, d\rangle\}) + \delta$ for all values $d \in \mathcal{D}_{v_\delta}$. And finally, for all non-singleton conjunctions $c \in (\mathcal{C} \setminus \mathcal{C}_1)$, $w(c) := \hat{w}(\{\langle \pi_c, 1\rangle\}) - \hat{w}(\{\langle \pi_c, 0\rangle\})$.

Let $s \in \mathcal{S}$ be any state. Since s defines exactly one value for v_δ, and $\mathcal{C}$ contains $\{\langle v_\delta, d\rangle\}$ for all values d of v_δ, we hence obtain $h^{\text{Pot}}_{\mathcal{C},w}(s) = \sum_{c \in \mathcal{C}, c \subseteq s} w(c)$

$$= \delta + \sum_{c \in \mathcal{C}_1, c \subseteq s} \hat{w}(c) + \sum_{c \in \mathcal{C} \setminus \mathcal{C}_1, c \subseteq s} (\hat{w}(\langle \pi_c, 1\rangle) - \hat{w}(\langle \pi_c, 0\rangle))$$

$$= \sum_{c \in \mathcal{C}_1, c \subseteq s} \hat{w}(c) + \sum_{c \in \mathcal{C} \setminus \mathcal{C}_1, c \subseteq s} \hat{w}(\langle \pi_c, 1\rangle) + \sum_{c \in \mathcal{C} \setminus \mathcal{C}_1, c \not\subseteq s} \hat{w}(\langle \pi_c, 0\rangle)$$

$$= h^{\text{Pot}}_{\hat{\mathcal{C}},\hat{w}}(s^{\mathcal{C}})$$

□

Since every feasible solution $\hat{w}$ to Pot$[\Pi^{\mathcal{C}}_{\text{TNF}}]$ gives a consistent and goal-aware potential heuristic for $\Pi^{\mathcal{C}}$, we can hence conclude from Lemma 1 that the corresponding w gives a consistent and goal-aware potential heuristic for Π.

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 though 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 (4) 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.

Example 4. *Consider the task with binary variables* v_1 *and* v_2*; initial state* $s_{\mathcal{I}} = \{\langle v_1, 0\rangle, \langle v_2, 0\rangle\}$*; goal* $s_* = \{\langle v_1, 1\rangle, \langle v_2, 1\rangle\}$*; and two actions* a_1 *which changes the value of* v_1 *from* 0 *to* 1*, and* a_2 *which changes the value of* v_2 *from* 0 *to* 1*. Consider the conjunctions* $c_1 = \{\langle v_1 = 0, v_2 = 0\rangle\}$*,* $c_2 = \{\langle v_1 = 0, v_2 = 1\rangle\}$*, and* $c_3 = \{\langle v_1 = 1, v_2 = 0\rangle\}$*. Assume that* $\mathcal{C}$ *contains all singleton conjunctions in addition to* c_1*,* c_2*, and* c_3*. Consider the potential heuristic with weights* $w(\{\langle v_1, 0\rangle\}) = w(\{\langle v_2, 0\rangle\}) = 2$ *and* $w(c_1) = -2$ *and* $w(c_2) = w(c_3) = -1$ *and* $w(\{p\}) = 0$ *for all other facts* p*. Obviously,* $h^{\text{Pot}}_{\mathcal{C},w}$ *is consistent and goal-aware. However, it is not possible to find* $\hat{w}$ *such that* $\hat{w}(\{\langle \pi_{c_2}, 1\rangle\}) - \hat{w}(\{\langle \pi_{c_2}, 0\rangle\}) = w(c_2)$ *and* $\hat{w}$ *satisfies Equation* (4) *for* $a_1^{\emptyset}$*, respectively Equation* (4) *for* $a_2^{\emptyset}$ *and* c_3*.*

While not every weight function w, inducing a consistent and goal-aware potential heuristic for Π, satisfies the constraints in Pot$[\Pi^{\mathcal{C}}_{\text{TNF}}]$, it remains an open question whether for every such w, there exists $\hat{w}$ that satisfies Pot$[\Pi^{\mathcal{C}}_{\text{TNF}}]$ and $h^{\text{Pot}}_{\mathcal{C},w} = h^{\text{Pot}}_{\hat{\mathcal{C}},\hat{w}}$. In Example 4, such $\hat{w}$ is given by $\hat{w}(\{\langle v_1, 0\rangle\}) = \hat{w}(\{\langle v_2, 0\rangle\}) = 1$ and $\hat{w}(\cdot) = 0$ otherwise.

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: $\mathrm{Pot}[\Pi]$ 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 $\mathrm{Pot}[\Pi, U]$.

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 obtained from any solution to $\mathrm{Pot}[\Pi^{\mathcal{C}}_{\mathrm{TNF}}, U]$ optimal for (O2), $h^{\mathrm{Pot}}_{\mathcal{C},w}(s) = h^*(s)$ 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. $\mathrm{Pot}[\Pi^{\mathcal{C}}_{\mathrm{TNF}}, U]$ then boils down to an LP encoding of paths in the state space of Π, with Equation (4) 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. □

Appendix A contains the full proof.

A simple trick now suffices to obtain h^* globally. We pessimistically interpret the cut-off U as a dead-end indicator, defining $h^{\mathrm{Pot}}_{\mathcal{C},w,U}(s) := h^{\mathrm{Pot}}_{\mathcal{C},w}(s)$ if $h^{\mathrm{Pot}}_{\mathcal{C},w}(s) < U$ and $h^{\mathrm{Pot}}_{\mathcal{C},w,U}(s) := \infty$ 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 corresponding to a solution to $\mathrm{Pot}[\Pi^{\mathcal{C}}_{\mathrm{TNF}}, U^*]$ optimal for (O2), $h^{\mathrm{Pot}}_{\mathcal{C},w,U^*} = h^*$.*

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 $h^{\mathrm{Pot}}_{\mathcal{C},w,U^*}$ that converges to $h^*[\Pi_0]$ as per Corollary 2. We will denote potential heuristics constructed this way as $u^{\mathrm{Pot}}_{\mathcal{C},w}$.

4.3 Relation to the State Equation

Pommerening et al. [2015] have shown that $\mathrm{Pot}[\Pi]$ 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 $h^{\mathrm{SEQ}}(s)$. In fact, there exist tasks and conjunction sets where *no* potential heuristic $h^{\mathrm{Pot}}_{\mathcal{C},w}$ equals $h^{\mathrm{SEQ}}[\Pi^{\mathcal{C}}]$ on all states.

Example 5. *Consider the task with variables v_1 and v_2 and domains $\mathcal{D}_{v_1} = \mathcal{D}_{v_2} = \{0, 1, 2\}$; goal $s_* = \{\langle v_1 = 2\rangle, \langle v_2 = 2\rangle\}$; and two actions: a_1 which changes the values of both v_1 and v_2 from 0 to 2, and a_2 which changes their values from 1 to 2. Both actions have 0-cost. The initial state is not important. Every state which does not assign both v_1 and v_2 to the same value is a dead-end. Consider the set of all singleton conjunctions $\mathcal{C}$. In order to satisfy the state equation constraints for $\langle v_1 = 2\rangle$ respectively $\langle v_2 = 2\rangle$ in any non-goal state, one of a_1 and a_2 must be selected. However, since a_1 consumes $\langle v_1 = 0\rangle$ and $\langle v_2 = 0\rangle$, and a_2 consumes $\langle v_1 = 1\rangle$ and $\langle v_2 = 1\rangle$, there exist feasible solutions to* Seq *only for the non dead-end states, i. e., h^{SEQ} recognizes all dead-ends.*

Consider the dead-ends $s_1 = \{\langle v_1 = 0\rangle, \langle v_2 = 1\rangle\}$ and $s_2 = \{\langle v_1 = 1, v_2 = 0\rangle\}$. Observe that there is no consistent and goal-aware dead-end potential heuristic $u^{\mathrm{Pot}}_{\mathcal{C},w}$ which recognizes both s_1 and s_2. In order for $u^{\mathrm{Pot}}_{\mathcal{C},w}$ to be consistent, it must hold $w(\langle v_1 = 0\rangle) + w(\langle v_2 = 0\rangle) \leq 0$ and $w(\langle v_1 = 1\rangle) + w(\langle v_2 = 1\rangle) \leq 0$. In order that $u^{\mathrm{Pot}}_{\mathcal{C},w}$ recognizes both s_1 and s_2, it must hold that $w(\langle v_1 = 0\rangle) + w(\langle v_2 = 1\rangle) > 0$ and $w(\langle v_1 = 1\rangle) + w(\langle v_2 = 0\rangle) > 0$. There obviously does not exist w that satisfies all 4 inequalities. The example can be easily extended to potential heuristics $h^{\mathrm{Pot}}_{\mathcal{C},w}$ in general.

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:

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

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

$\mathcal{C}_\mathsf{S}$Pot : Similar to $\mathcal{C}_\mathsf{S}$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 $h^{\mathrm{SEQ}}[\Pi^\mathcal{C}]$; but now we also add a new potential heuristic $u^{\mathrm{Pot}}_{\mathcal{C},w}$ (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 $h^{\mathrm{SEQ}}[\Pi^\mathcal{C}](s) \neq \infty$. We need to extend $\mathcal{C}$ to $\mathcal{C}' \supseteq \mathcal{C}$ such that $h^{\mathrm{SEQ}}[\Pi^{\mathcal{C}'}](s) = \infty$. We do so by iteratively finding a conjunction $x \notin \mathcal{C}$ whose SEQ constraint is not satisfied by some LP solution. We set $\mathcal{C} := \mathcal{C} \cup \{x\}$. If $h^{\mathrm{SEQ}}[\Pi^\mathcal{C}](s) = \infty$, we stop; else, we iterate. By Corollary 1, the termination condition must hold eventually.

It remains to show how to choose x. Denote by Seq$[\Pi^\mathcal{C}]$ the LP underlying $h^{\mathrm{SEQ}}[\Pi^\mathcal{C}](s)$. The refinement is based on a concrete solution Count to 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., $pre_{a_s} = \emptyset$, $e\!f\!f_{a_s} = s^\mathcal{C}$, $pre_{a_*} = s_*^\mathcal{C}$, and $e\!f\!f_{a_*} = \emptyset$. We denote by $\mathcal{A}^\mathcal{C}_{\mathsf{Count}}$ the actions of (1) and (2). For any (partial) variable assignment P over the variables of $\Pi^\mathcal{C}$, we denote by $P|_\mathcal{V}$ the projection of P onto $\mathcal{V}$. 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* $a^C \in \mathcal{A}^\mathcal{C}_{\mathsf{Count}}$, $\mathsf{regr}(pre_{a_0^{C_0}}, a) \not\subseteq pre_{a^C}$. (Recall that the actions in $\Pi^\mathcal{C}$ represent action occurrences in the original task, carrying the context of application.)

In general, not every LP solution Count must actually select an action satisfying (*). However, it is always guaranteed that there is at least one optimal LP solution for which the desired $a_0^{C_0}$ exists. We next show how such LP solution is obtained starting from an arbitrary LP solution Count. If $a_0^{C_0}$ already exists for the considered LP solution, we can immediately proceed with the selection of x (as shown below). If there is no such $a_0^{C_0}$, we construct from Count another feasible LP solution Count' which satisfies (*) as follows. Consider the graph with nodes $\mathcal{A}^\mathcal{C}_{\mathsf{Count}}$ and edges $a^C \rightarrow a_0^{C_0}$ 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 $\mathsf{regr}(pre_{a_1}, a_n) \subseteq pre_{a_n}$.

Observe that, along this sequence, every fact of $\Pi^\mathcal{C}$ is produced as often as it is consumed. For the original facts from Π, this follows immediately from the TNF assumption. Moreover, given that Π is in TNF, it holds for every $\langle \pi_c, 1\rangle$ and every a_i with $\langle \pi_c, 1\rangle \in e\!f\!f_{a_i}$ that $\langle \pi_c, 0\rangle \in pre_{a_i}$. Due to the selection of the actions $a_1, \dots, a_n$ and due to the definition of actions in $\Pi^\mathcal{C}$, $\langle \pi_c, 0\rangle \in e\!f\!f_{a_i}$ implies $\langle \pi_c, 1\rangle \in pre_{a_i}$. In conclusion, every π_c-fact is indeed produced as often as it is consumed.

But this means that 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: let $\gamma := \min_{1 \leq i \leq n} \mathsf{Count}_{a_i}$; set $\mathsf{Count}'_{a^C} = \mathsf{Count}_{a^C}$ for every action occurrence a^C such that $a^C \neq a_i$ for all $1 \leq i \leq n$, and set $\mathsf{Count}_{a_i} := \mathsf{Count}_{a_i} - \gamma$. The explanation above implies that, for every fact of $\Pi^\mathcal{C}$, the the production part is reduced by the exact same amount as the consumption part, i. e., Count' does still satisfies all SEQ constraints. However, it holds $\mathsf{Count}'_{a_i} = 0$ for at least one action of the sequence $a_1, \dots, a_n$. 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 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 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.

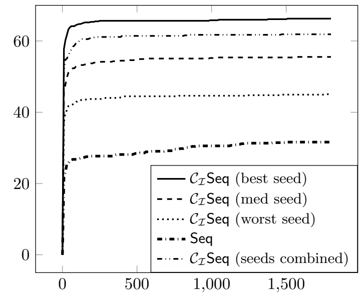

Figure 2: Coverage (in %) over time (in s) for different variable orders in conjunction generation, see text.

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 $M \in \{2, 4, 8, \dots, 1024, \infty\}$, 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 $\mathcal{C}_{\mathcal{I}}\mathsf{Seq}$ and $x = \infty$, for five randomly generated orders, picking the per-instance best, median, and worst variable order. The variance in coverage is large. (We can also see that $\mathcal{C}_{\mathcal{I}}\mathsf{Seq}$ solves an instance either quickly or not at all.)

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.

Domain	#	Blind	h^1	h^C	PDB	Seq	Pot	$\mathcal{C}_S\mathsf{Seq}$ 128	$\mathcal{C}_S\mathsf{Seq}$ 256	$\mathcal{C}_S\mathsf{Seq}$ ∞	$\mathcal{C}_S\mathsf{Pot}$ 128	$\mathcal{C}_S\mathsf{Pot}$ 256	$\mathcal{C}_S\mathsf{Pot}$ ∞	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). $h^{\mathcal{C}}$: 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.

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 $\mathcal{C}_{\mathcal{I}}\mathsf{Seq}$ 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.

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.

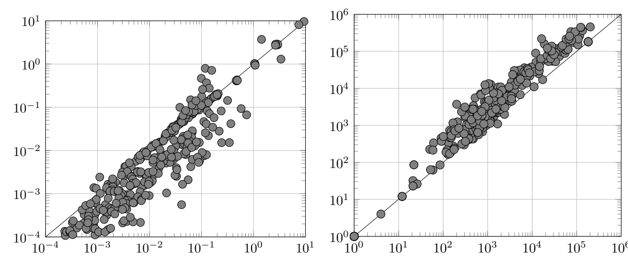

Figure 3: Per-state runtime (left) and number of visited states (right), of $\mathcal{C}_S\mathsf{Seq}$ (x-axes) vs. $\mathcal{C}_S\mathsf{Pot}$ (y-axes).

$\mathcal{C}_S\mathsf{Seq}$ 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 Steve Chien, Minh Do, Alan Fern, and Wheeler Ruml, editors, *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS'14)*, pages 47–55. AAAI Press, 2014.

[Bonet, 2013] Blai Bonet. An admissible heuristic for SAS+ planning obtained from the state equation. In Francesca Rossi, editor, *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*, pages 2268–2274. AAAI Press/IJCAI, 2013.

[Haslum, 2009] Patrik Haslum. $h^m(P) = h^1(P^m)$: Alternative characterisations of the generalisation from $h^{\max}$ to h^m. In Alfonso Gerevini, Adele Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, pages 354–357. AAAI Press, 2009.

[Haslum, 2012] Patrik Haslum. Incremental lower bounds for additive cost planning problems. In Blai Bonet, Lee McCluskey, José Reinaldo Silva, and Brian Williams, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, pages 74–82. AAAI Press, 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 Blai Bonet, Lee McCluskey, José Reinaldo Silva, and Brian Williams, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, pages 181–189. AAAI Press, 2012.

[Pommerening and Helmert, 2015] Florian Pommerening and Malte Helmert. A normal form for classical planning tasks. In Ronen Brafman, Carmel Domshlak, Patrik Haslum, and Shlomo Zilberstein, editors, *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, pages 188–192. AAAI Press, 2015.

[Pommerening *et al.*, 2014] Florian Pommerening, Gabriele Röger, Malte Helmert, and Blai Bonet. LP-based heuristics for cost-optimal planning. In Steve Chien, Minh Do, Alan Fern, and Wheeler Ruml, editors, *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS'14)*, pages 226–234. AAAI Press, 2014.

[Pommerening *et al.*, 2015] Florian Pommerening, Malte Helmert, Gabriele Röger, and Jendrik Seipp. From non-negative to general operator cost partitioning. In Blai Bonet and Sven Koenig, editors, *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, pages 3335–3341. AAAI Press, January 2015.

[Pommerening *et al.*, 2017] Florian Pommerening, Malte Helmert, and Blai Bonet. Higher-dimensional potential heuristics for optimal classical planning. In Satinder Singh and Shaul Markovitch, editors, *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*, pages 3636–3643. AAAI Press, February 2017.

[Seipp *et al.*, 2015] Jendrik Seipp, Florian Pommerening, and Malte Helmert. New optimization functions for potential heuristics. In Ronen Brafman, Carmel Domshlak, Patrik Haslum, and Shlomo Zilberstein, editors, *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, pages 193–201. AAAI Press, 2015.

[Seipp *et al.*, 2016a] Jendrik Seipp, Florian Pommerening, Gabriele Röger, and Malte Helmert. Correlation complexity of classical planning domains. In Subbarao Kambhampati, editor, *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'16)*, pages 3242–3250. AAAI Press/IJCAI, 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.

[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 Christian Bessiere, editor, *Proceedings of the Thirteenth International Conference on Principles and Practice of Constraint Programming (CP'07)*, volume 4741 of *Lecture Notes in Computer Science*, pages 651–665. Springer-Verlag, 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 *ICAPS 2007 Workshop on Heuristics for Domain-Independent Planning: Progress, Ideas, Limitations, Challenges (HDIP'07)*, pages 20–24, 2007.

A Proofs

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

Proof. Let Seq$[\Pi^{\mathcal{C}}]$ be the LP underlying $h^{\mathrm{SEQ}}[\Pi^{\mathcal{C}}](s)$, and Seq$\mathcal{C}$ that underlying $h^{\mathcal{C}\mathrm{SEQ}}(s)$. If there is no feasible solution to Seq$[\Pi^{\mathcal{C}}]$, then $h^{\mathrm{SEQ}}[\Pi^{\mathcal{C}}](s) = \infty$, and $h^{\mathrm{SEQ}}[\Pi^{\mathcal{C}}](s) \geq h^{\mathcal{C}\mathrm{SEQ}}(s)$ follows trivially. For the rest of proof, we assume X to be a feasible solution for Seq$[\Pi^{\mathcal{C}}]$, where X_{a^C} denotes the count-value for the action occurrence $a^C \in \mathcal{A}^{\mathcal{C}}$.

We next show how to construct from X a feasible solution Y for Seq$\mathcal{C}$ with equal objective value. We choose the values of Y_a and $Y_a^{x \to x'}$ in the following way. For every action $a \in \mathcal{A}$, we set

$$Y_a := \sum_{C \subseteq \mathcal{C} : a^C \in \mathcal{A}^{\mathcal{C}}} X_{a^C} \qquad \text{(T1.1)}$$

(all a^C over the same base action a). The transition variables are defined as follows:

- For every transition $c \xrightarrow{a} c'$:

$$Y_a^{c \to c'} := \sum_{\substack{C \subseteq \mathcal{C} : a^C \in \mathcal{A}^C, \\ \langle \pi_c, 0 \rangle \in \mathit{eff}_{a^C}, \\ \langle \pi_{c'}, 1 \rangle \in \mathit{eff}_{a^C}}} X_{a^C} \qquad \text{(T1.2)}$$

- For every transition $\top \xrightarrow{a} c$:

$$Y_a^{\top \to c} := \sum_{\substack{C \subseteq \mathcal{C} : a^C \in \mathcal{A}^C, \\ \langle \pi_c, 1 \rangle \in \mathit{eff}_{a^C}}} X_{a^C} \qquad \text{(T1.3)}$$

- For every transition $c \xrightarrow{a} \top$:

$$Y_a^{c \to \top} := \sum_{\substack{C \subseteq \mathcal{C} : a^C \in \mathcal{A}^C, \\ \langle \pi_c, 1 \rangle \in \mathit{pre}_{a^C}, \\ \langle \pi_c, 0 \rangle \in \mathit{eff}_{a^C}}} X_{a^C} \qquad \text{(T1.4)}$$

We start with showing that Equation (1) is satisfied for all facts $p = \langle v, d \rangle$ with $v \in \mathcal{V}$. Denote by $\mathit{Prod}[\Pi^{\mathcal{C}}]$ and $\mathit{Cons}[\Pi^{\mathcal{C}}]$ the producers and consumers defined in $\Pi^{\mathcal{C}}$. It is easy to see that $\mathit{Prod}[\Pi^{\mathcal{C}}](p) = \{a^C \mid a \in \mathit{Prod}(p), C \subseteq \mathcal{C} \text{ s.t. } a^C \in \mathcal{A}^{\mathcal{C}}\}$ and that $\mathit{Cons}[\Pi^{\mathcal{C}}](p) = \{a^C \mid a \in \mathit{Cons}(p), C \subseteq \mathcal{C} \text{ s.t. } a^C \in \mathcal{A}^{\mathcal{C}}\}$. Given (T1.1), it holds that $\sum_{a \in \mathit{Prod}(p)} Y_a = \sum_{a \in \mathit{Prod}(p)} \sum_{C \subseteq \mathcal{C}, a^C \in \mathcal{A}^C} X_{a^C} = \sum_{a^C \in \mathit{Prod}[\Pi^{\mathcal{C}}](p)} X_{a^C}$, and similarly for the consumers. Since X satisfies the constraint for p in Seq$[\Pi^{\mathcal{C}}]$ by assumption, Y hence satisfies p's constraint in $\mathcal{C}$Seq.

Consider next Equation (3) for any partial variable merge with associated variables V. We first show that every action occurrence $a^C \in \mathcal{A}^{\mathcal{C}}$ counts towards at most one transition in this partial variable merge. Assume the contrary, and let $a^C \in \mathcal{A}^{\mathcal{C}}$ be an action occurrence where this is not true. Let $x_1 \xrightarrow{a} x_2$ and $x_3 \xrightarrow{a} x_4$ be two state-changing transitions in the considered variable merge which both count X_{a^C}. Note that $x_1 \neq \top$ or $x_2 \neq \top$, and $x_3 \neq \top$ or $x_4 \neq \top$. We distinguish between the following cases

- $x_2 = c_1$ and $x_4 = c_2$ for $c_1, c_2 \in \mathcal{C}$ with $\mathcal{V}(c_1) = \mathcal{V}(c_2) = V$ and $c_1 \neq c_2$. From (T1.2) and (T1.3) and the definition of a^C, it follows that $c_1, c_2 \in C$ what contradicts the compatibility requirement of C.
- $x_1 = c_1$ and $x_3 = c_2$ for $c_1, c_2 \in \mathcal{C}$ with $\mathcal{V}(c_1) = \mathcal{V}(c_2) = V$ and $c_1 \neq c_2$. Note that both (T1.2) and (T1.4) imply that $\langle \pi_c, 1 \rangle \in \mathit{pre}_{a^C}$ (c being the conjunction in the respective conditions). The latter directly imposes this as condition. Regarding the former, (T1.2), since Π is in TNF and $\langle \pi_{c'}, 1 \rangle \in \mathit{eff}_{a^C}$, it follows that $\mathcal{V}(c') \subseteq \mathcal{V}(\mathsf{regr}(c', a)) \subseteq \mathcal{V}(\mathit{pre}_{a^C})$. From $\langle \pi_c, 0 \rangle \in \mathit{eff}_{a^C}$ it follows in particular that $c || \mathit{pre}_{a^C}$, so since $\mathcal{V}(c) = \mathcal{V}(c')$ it holds that $c \subseteq \mathit{pre}_{a^C}$, i.e., $\langle \pi_c, 1 \rangle \in \mathit{pre}_{a^C}$. Hence, $\langle \pi_{c_1}, 1 \rangle \in \mathit{pre}_{a^C}$ and $\langle \pi_{c_2}, 1 \rangle \in \mathit{pre}_{a^C}$, what is in contradiction to the compatibility of C.
- $x_2 = c_1$ and $x_3 = c_2$ for $c_1, c_2 \in \mathcal{C}$ with $\mathcal{V}(c_1) = \mathcal{V}(c_2) = V$ where either $x_1 \neq c_2$ or $x_4 \neq c_1$. From (T1.2), (T1.3), and (T1.4), it follows that $\langle \pi_{c_2}, 0 \rangle \in \mathit{eff}_{a^C}$ and $\langle \pi_{c_1}, 1 \rangle \in \mathit{eff}_{a^C}$. By definition of a^C, $\langle \pi_{c_2}, 0 \rangle \in \mathit{eff}_{a^C}$ implies that $c_2 || \mathit{pre}_{a^C}$. Therefore, the partial variable merge must contain the transition $c_2 \xrightarrow{a} c_1$. Since Π is in TNF, a can only label a single transition going into c_1, respectively a single transition going out of c_2. But this means that $x_1 = c_2$ and $x_4 = c_1$, a contradiction to the assumption.

Since $x_1 \xrightarrow{a} x_2$ and $x_3 \xrightarrow{a} x_4$ are two different transitions, one of the three cases must apply. But all of them lead to a contradiction to one of the assumptions. This hence shows that every a^C is associated with at most one transition per partial variable merge. Moreover, if Equation (3) requires only $\leq$, it follows immediately that the respective constraint is satisfied. If Equation (3) for action $a \in \mathcal{A}$ and partial variable merge over variables V enforces equality, then $V \subseteq \mathcal{V}(\mathit{pre}_a)$ and there must exist $c \in \mathcal{C}$ with $\mathcal{V}(c) = V$ and either (a) $c \subseteq \mathit{pre}_a$ and $c \not\| \mathit{eff}_a$, or (b) $\mathit{pre}_a \subseteq \mathsf{regr}(c, a)$. For (a) it follows for all action occurences a^C of a that $\langle \pi_c, 1 \rangle \in \mathit{pre}_{a^C}$ and $\langle \pi_c, 0 \rangle \in \mathit{eff}_{a^C}$. Let $c \xrightarrow{a} x$ be the corresponding transition. If $x = \top$, then $Y_a^{c \to \top} = \sum_{\substack{C \subseteq \mathcal{C}, a^C \in \mathcal{A}^C, \\ \langle \pi_c, 1 \rangle \in \mathit{pre}_{a^C}, \\ \langle \pi_c, 0 \rangle \in \mathit{eff}_{a^C}}} X_{a^C} = \sum_{C \subseteq \mathcal{C}, a^C \in \mathcal{A}^C} = Y_a$. If $x = c'$ for some $c' \in \mathcal{C}$, then $\mathit{pre}_a \subseteq \mathsf{regr}(c', a)$, and hence $\langle \pi_{c'}, 1 \rangle \in \mathit{eff}_{a^C}$ for every action occurrence a^C of a (Condition (2) of $\Pi^{\mathcal{C}}$ definition). In other words, $Y_a^{c \to c'} = \sum_{C \subseteq \mathcal{C}, a^C \in \mathcal{A}^C} X_{a^C} = Y_a$. For (b), let $x \xrightarrow{a} c$ be the corresponding transition. Due to condition (2) of the $\Pi^{\mathcal{C}}$ definition, it holds that $\langle \pi_c, 1 \rangle \in \mathit{eff}_{a^C}$ for all action occurences a^C with base action a. If $x = \top$, then $Y_a^{\top \to c} = \sum_{\substack{C \subseteq \mathcal{C}, a^C \in \mathcal{A}^C, \\ \langle \pi_c, 1 \rangle \in \mathit{eff}_{a^C}}} X_{a^C} = \sum_{C \subseteq \mathcal{C}, a^C \in \mathcal{A}^C} X_{a^C} = Y_a$. The case $x = c'$ follows from the arguments given before. Thus, Y satisfies Equation (3) regardless of whether $\leq$ or $=$ is enforced.

It is left to show that Y satisfies Equation (2) for all conjunctions $c \in \mathcal{C}$. Let $c \in \mathcal{C}$ be arbitrary. We show that (a) the production part for c in Equation (2) is at least as large as the production part for $\langle \pi_c, 1 \rangle$ in Equation (1), and that (b) the consumption part for c is at most as large as the consumption part of $\langle \pi_c, 1\rangle$. As shown above, every action occurrence is counted in at most one transition per variable merge. Moreover, as before, for every a^C, c, and c' in (T1.2), it holds that $\langle \pi_c, 1\rangle \in pre_{a^C}$. Therefore, it follows from (T1.2), (T1.4), and the definition $Cons(\langle \pi_c, 1\rangle)$ that (b) is satisfied. Regarding (a), observe that every action occurrence a^C with $\langle \pi_c, 1\rangle \in \mathit{eff}_{a^C}$ is counted in at least one transition $x \xrightarrow{a} c$. Let a^C be any such action occurrence. By definition, $\mathsf{regr}(c, a) \neq \bot$, and hence a must induce at least one transition going into c in the respective variable merge. If $\top \xrightarrow{a} c$, then $Y_a^{\top\to c}$ counts X_{a^C} by construction (T1.3). If $c' \xrightarrow{a} c$, then $c' || \mathsf{regr}(c, a)$ and $c' \neq c$ by definition of regression, i. e., $c' \not\parallel \mathit{eff}_a$. Thus, $\langle \pi_{c'}, 0\rangle \in \mathit{eff}_{a^C}$, so (T1.2) makes sure that X_{a^C} is counted in $Y_a^{c'\to c}$. Since $\Delta_c = \Delta_{\langle \pi_c, 1\rangle}$, we conclude that Y also satisfies Equation (2) for all conjunctions $c \in \mathcal{C}$.

Finally, note that (T1.1) implies that that the objective value of Y is the same as that of X. This completes the proof. □

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

Proof. Consider the following transportation example. The map consists of two locations A and B, and is fully connected. There is a single truck t with load capacity l, which must bring n packages $p_1, \dots, p_n$ to their destinations. To do so, there are three types of actions: to $move$ the truck between A and B; to $load$ package p_i into truck at B, requiring that enough load capacity is available; and to $unload$ the package p_i at location A. All actions have cost 1. In the initial state, t is at A, l is 1, and all packages are at B. The goal is to have all variables $t, p_1, \dots, p_n$ at A, and $l = 1$. Every optimal plan for this task needs to do one $load$, one $unload$, and two $move$ actions for every package, summing up to a total of $h^*(s_\mathcal{I}) = 4n$.

In $h^{\mathrm{SEQ}}[\Pi^{\mathcal{C}}]$, considering all conjunctions c of size $|c| \leq 3$ makes visible that no two packages can be in the truck at the same time, yielding $h^{\mathrm{SEQ}}[\Pi^{\mathcal{C}}](s_\mathcal{I}) = h^*(s_\mathcal{I})$. Similar to Example 1, every solution to $\mathrm{Seq}[\Pi^{\mathcal{C}}]$ must *load* and *unload* every package once. To see that $h^{\mathrm{SEQ}}[\Pi^{\mathcal{C}}](s_\mathcal{I})$ must also account for two $move$ actions for each package, consider the action $a_0 = unload(p_i, A)$ for any package p_i. Every action occurrence a_0^C of a_0 consumes the conjunction $c_i = \{\langle t = A\rangle, \langle p_i = T\rangle\}$. The only possibility to produce c_i is via an action occurrence of the $move$ action, moving the truck to A, and assuming $\langle p_i = T\rangle$ in its context. Observe that the same $move$ action occurrence cannot be used to achieve c_i and c_j for two different packages $i \neq j$. This is true because every action occurrence a_1^C of $a_1 = move(B, A)$ with $\{\langle p_i = T\rangle, \langle p_j = T\rangle\} \subseteq C$ consumes the conjunction $c' = \{\langle t = B\rangle, \langle p_i = T\rangle, \langle p_j = T\rangle\}$. However, c' cannot be produced without violating some constraint. There are two possibilities: (1) via an action occurrence of $move(A, B)$, including $\{\langle p_i = T\rangle, \langle p_j = T\rangle\}$ in the context; and (2) loading one of the packages, e. g., $load(p_i, B)$ including $\{\langle p_j = T\rangle\}$ in the context. Option (1) cannot be used, as this would basically lead to a cyclic dependency between the respective $move(A, B)$ and $move(B, A)$ action occurrences. Option (2) leads to the consumption of the conjunction $\{\langle p_j = T\rangle, \langle l = 1\rangle\}$, which obviously cannot be produced without violating the state equation constraints. Hence, for every package p_i, c_i must be produced through a separate $move(B, A)$ action occurrence. Since every such occurrence consumes $\langle t = B\rangle$, its state equation constraint forces to count for one $move(A, B)$ action application per package. This shows that $h^{\mathrm{SEQ}}[\Pi^{\mathcal{C}}](s_\mathcal{I}) = h^*(s_\mathcal{I})$.

In contrast, in order to obtain $h^{\mathcal{C}\mathrm{SEQ}}(s_\mathcal{I}) = h^*(s_\mathcal{I})$, $\mathcal{C}$ needs to contain exponentially many conjunctions. Let $\mathcal{C}$ be any set of conjunctions. Let Count denote any solution to $\mathrm{Seq}\mathcal{C}$ with minimal objective value. Consider first the state equation constraints in Equation (1) over facts $p = \langle v, d\rangle$. For $v = t$, the state equation constraints are satisfied if the number of $move(B, A)$ action counts matches the number of $move(A, B)$ action counts. For $v \neq t$, the $move$ count variables do not appear in any constraint. Next, consider any partial variable merge over the variable set V, and let m denote the number of packages considered in V. For $V = \{v\}$, the satisfaction of the corresponding constraints in Equation (2) and Equation (3) are implied by the satisfaction of the state equation constraints, Equation (1), for v. Assume that $|V| > 1$. We distinguish between the following cases: For $m = 0$, i. e., $V = \{t, l\}$, the abstract initial state in the corresponding partial variable merge is identical to the the abstract goal state. No $move$ action transitions are required to satisfy the constraints corresponding to V. For $m > 0$ but $t \notin V$, the corresponding partial variable merge does not contain any $move$ transition. For $m > 0$, $t \in V$, but $l \notin V$, to reach the abstract goal state from the abstract initial state, at most one $move(B, A)$ transition and at most one $move(A, B)$ transition is required to satisfy the constraints. If $m > 0$ and $t, l \in V$, reaching the abstract goal state requires at most m $move(B, A)$ transitions, and at most $2m$ $move$ transitions in total. Choosing $\mathsf{Count}_{move(A,B)}$ or $\mathsf{Count}_{move(B,A)}$ to a value larger than the maximal number of $move(A, B)$ and $move(B, A)$ transition counts over all considered partial variable merges, leads to a contradiction to the minimality of Count. Hence, the combination of all the above cases shows that $\mathsf{Count}_{move(A,B)} + \mathsf{Count}_{move(B,A)} \leq \max\{2, 2\hat{m}\}$, for the variable set $\hat{V}$ with $t, l \in \hat{V}$ and number of packages $\hat{m}$ maximal among all such variable sets. As an immediate consequence, if it holds that $h^{\mathcal{C}\mathrm{SEQ}}(s_\mathcal{I}) = h^*(s_\mathcal{I})$, then it must also hold that $\hat{m} = n$, i. e., $\hat{V} = \mathcal{V}$.

We finally show that if $\mathcal{C}$ contains from any optimal plan less than $4n - 3$ states, then the partial variable merge corresponding to $\mathcal{V}$ requires less than n $move(B, A)$ transitions to reach the abstract goal, and hence $h^{\mathcal{C}\mathrm{SEQ}}(s_\mathcal{I})$ cannot encode $h^*(s_\mathcal{I})$. Since there are exponentially many optimal plans, one for each permutation of $p_1, \dots, p_n$, and they all commonly visit exactly two states (the initial state and the goal state), this hence shows that $\mathcal{C}$ must contain exponentially many conjunctions. We show the claim by contraposition. Assume there is an optimal plan $\pi = \langle a_1, \dots, a_{4n}\rangle$ visiting states $s_\mathcal{I} = s_0, s_1, \dots, s_{4n}$ with indices $0 \leq i < j \leq 4n$ such that $i + 3 \leq j$ and $s_i, s_j \notin \mathcal{C}$. Since π is optimal, one of the actions $a_i, \dots, a_{i+3}$ must be $move(B, A)$. However, since $s_i, s_j \notin \mathcal{C}$, both states are represented in the partial variable merge by the same abstract state, introducing a shortcut, and avoiding at least one $move(B, A)$ transition. Hence, the minimal number of $move(B, A)$ transitions required to reach the abstract goal must be smaller than n.

In conclusion, $\mathcal{C}'$ must contain exponentially many conjunctions (in n) in order that $h^{\mathcal{C}'\text{SEQ}}(s_\mathcal{I}) \geq h^{\text{SEQ}}[\Pi^\mathcal{C}](s_\mathcal{I}) = h^*(s_\mathcal{I})$, while $|\mathcal{C}|$ is polynomially bounded in n.

□

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 obtained from any solution to* $\text{Pot}[\Pi^\mathcal{C}_\text{TNF}, U]$ *optimal for (O2), $h^{\text{Pot}}_{\mathcal{C},w}(s) = h^*(s)$ for all states s with $h^*(s) \leq U$.*

Proof. We will first show that the claim holds for $\mathcal{C} := \mathcal{S}$, i. e., the set of all states in the task, and $\hat{\mathcal{C}} := \{\langle \pi_s, 1\rangle \mid s \in \mathcal{S}\}$. We will then extend the arguments, showing that the claim still holds if we consider in $\mathcal{C}$ additionally all singleton conjunctions over Π's facts, and in $\hat{\mathcal{C}}$ all singleton conjunctions over $\Pi^\mathcal{C}_\text{TNF}$'s facts, i. e., matching the requirements of Theorem 4.

Consider the weight function $\hat{w} : \hat{\mathcal{C}} \mapsto \mathbb{R}$ where $\hat{w}(\{\langle \pi_s, 1\rangle\}) := \min\{h^*(s), U\}$. Observe that $\hat{w}$ satisfies all constraints of $\text{Pot}[\Pi^\mathcal{C}_\text{TNF}, U]$ over the conjunctions $\hat{\mathcal{C}}$: $\hat{w}(s)$ is bounded by U for all $s \in \mathcal{C}$ by construction. Equation (5) is satisfied since $\hat{w}(s_*) = h^*(s_*) = 0$. Equation (4) is satisfied since every occurrence a^C can consume, respectively produce, at most one $\langle \pi_s, 1\rangle$. If a^C does not produce any $\langle \pi_s, 1\rangle$, then since $\hat{w}(s) \geq 0$ for all $s \in \mathcal{C}$, the constraint is satisfied. If a^C produces some $\langle \pi_s, 1\rangle$, then since Π is in TNF, a^C must also consume some $\langle \pi_{s'}, 1\rangle$. It holds that $s' = \mathsf{regr}(s, a)$, i. e., $s = s'[\![a]\!]$. Equation (4) turns into $\hat{w}(\{\langle \pi_{s'}, 1\rangle\}) - \hat{w}(\{\langle \pi_s, 1\rangle\}) \leq \mathsf{cost}_a$, which by definition of $\hat{w}$ is equivalent to $h^*(s') - h^*(s) \leq \mathsf{cost}_a$. In conclusion, $\hat{w}$ satisfies all constraints of $\text{Pot}[\Pi^\mathcal{C}_\text{TNF}, U]$. Finally, observe that $\hat{w}$ is the only possibility to maximize (O2). Assume there was a different $\hat{w}'$ maximizing (O2). There must be some $s \in \mathcal{C}$ such that $\hat{w}'(\{\langle \pi_s, 1\rangle\}) > \hat{w}(\{\langle \pi_s, 1\rangle\}) = h^*(s)$. As action occurences of $\Pi^\mathcal{C}_\text{TNF}$ enumerate all possible transitions in Π, the satisfaction of the consistency constraints imply that $\hat{w}'$ violates either $\hat{w}'(\{\langle \pi_{s_*}, 1\rangle\}) = 0$ or $\hat{w}'(\{\langle \pi_s, 1\rangle) \leq U$ for some $s \in \mathcal{S}$. This shows the claim.

The proof can be extended to $\hat{\mathcal{C}}$ containing *all* singleton conjunctions over $\Pi^\mathcal{C}_\text{TNF}$'s facts if the normalization of the weights of $\hat{\mathcal{C}}$ matches the frequency of conjunctions in Π. Denote by $\hat{freq}$ the normalization coefficients for $\hat{\mathcal{C}}$. We define $\hat{freq}(\{\langle v, d\rangle\}) := freq(\{\langle v, d\rangle\})$ for $v \in \mathcal{V}$; $\hat{freq}(\{\langle \pi_c, 1\rangle\}) := freq(c)$ and $\hat{freq}(\{\langle \pi_c, 0\rangle\}) := 1 - freq(c)$ and $\hat{freq}(\{\langle \pi_c, \top\rangle\}) := 0$. It is straightforward to show that maximizing the average heuristic value over all states in $\mathcal{S}$ is equivalent to maximizing $\sum_{\hat{c} \in \hat{\mathcal{C}}} \hat{w}(\hat{c}) \cdot \hat{freq}(\hat{c})$. Using the same arguments as before, we can show the existence of $\hat{w}$ satisfying $\text{Pot}[\Pi^\mathcal{C}_\text{TNF}, U]$ with corresponding w such that $h^{\text{Pot}}_{\mathcal{C},w}(s) = h^*(s)$ for all states s. Due to Lemma 1 and Equation (5), for every $\hat{w}'$ satisfying $\text{Pot}[\Pi^\mathcal{C}_\text{TNF}, U]$ and for every corresponding w', it holds that $h^{\text{Pot}}_{\mathcal{C},w'}(s) \leq h^*(s)$. Hence, $\hat{w}$ maximizes the objective, what concludes the proof. □