Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020)

Strengthening Potential Heuristics with Mutexes and Disambiguations

Daniel Fišer, Rostislav Horčík, Antonín Komenda
Czech Technical University in Prague,
Faculty of Electrical Engineering,
Prague, Czech Republic
danfis@danfis.cz, {xhorcik,antonin.komenda}@fel.cvut.cz

Abstract

Potential heuristics assign a numerical value (potential) to each fact and compute the heuristic value for a given state as the sum of these potentials. A mutex is an invariant stating that a certain combination of facts cannot be part of any reachable state. In this paper, we use mutexes to improve potential heuristics in two ways. First, we show that the mutex-based disambiguations of the goal and preconditions of operators leads to a less constrained linear program yielding stronger heuristics. Second, we utilize mutexes in a construction of new optimization functions based on counting of the number of states containing certain sets of facts. The experimental evaluation shows a significant increase in the number of solved tasks.

1 Introduction

The most common approach to solving classical planning problems is a heuristic search. In this paper, we focus on the family of admissible heuristics called *potential heuristics* (Pommerening et al. 2015a) that assign a numeric potential to each fact and the resulting heuristic value for a given state is computed as a sum of the potentials of the facts in the state. Pommerening et al. (2015a) showed that potential heuristics produce the same heuristic value for a given state as the state equation heuristic (van den Briel et al. 2007; Bonet 2013) if they are optimized for each state individually, but, in practice, the potentials for all facts are found only once before the search starts and then re-used during the search in a very fast evaluation of states.

Since their introduction, potential heuristics are continuously studied. Seipp, Pommerening, and Helmert (2015) introduced several new optimization functions that provide ways to select different sets of potentials in order to increase the heuristic values. Seipp et al. (2016) introduced a new complexity measure for classical planning tasks, called the correlation complexity, based on a study of a dimensionality of features of potential heuristics, i.e., extending potentials from single facts to sets of facts. Pommerening, Helmert, and Bonet (2017) provided a detailed description of how to construct potential heuristics for high-dimensional features.

We study possible impacts of inferred state invariants on the quality of potential heuristics. In particular, we consider state invariants, called *mutexes*, stating that a certain set of facts cannot be part of any reachable state (Bonet and Geffner 2001). Mutexes were studied in the context of regression planning (Alcázar et al. 2013), pruning of the unreachable or dead-end operators (Alcázar and Torralba 2015; Fišer and Komenda 2018; Fišer, Torralba, and Shleyfman 2019), SAT-based planning (Rintanen, Heljanko, and Niemelä 2006; Chen, Xing, and Zhang 2007; Huang, Chen, and Zhang 2012), or translation of STRIPS representation into FDR (Helmert 2009; Fišer and Komenda 2018).

We utilize mutexes in potential heuristics in two ways. First, we relax the constraints describing potential heuristics by using mutexes and so-called disambiguation (Alcázar et al. 2013) to infer which facts cannot be part of the goal states or states where operators are applied. This leads to higher (or at worst the same) heuristic values for all optimization functions, because the optimizer becomes less restricted in the search for the best potentials. Second, we use mutexes in new optimization functions aiming at more accurate estimations of the number of reachable states and thus improving average heuristic values over all reachable states.

2 Background

We consider the finite domain representation (FDR) of planning tasks (Bäckström and Nebel 1995). An **FDR planning task** Π is specified by a tuple $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$. $\mathcal{V}$ is a finite set of **variables**, each variable $V \in \mathcal{V}$ has a finite **domain** $\mathrm{dom}(V)$. A **fact** $\langle V, v \rangle$ is a pair of a variable $V \in \mathcal{V}$ and one of its values $v \in \mathrm{dom}(V)$. The set of all facts is denoted by $\mathcal{F} = \{\langle V, v \rangle \mid V \in \mathcal{V}, v \in \mathrm{dom}(V)\}$, and the set of facts of variable V is denoted by $\mathcal{F}_V = \{\langle V, v \rangle \mid v \in \mathrm{dom}(V)\}$. A **partial state** p is a variable assignment over some variables $\mathrm{vars}(p) \subseteq \mathcal{V}$. We write $p[V]$ for the value assigned to the variable $V \in \mathrm{vars}(p)$ in the partial state p. We also identify p with the set of facts contained in p, i.e., $p = \{\langle V, p[V] \rangle \mid V \in \mathrm{vars}(p)\}$. A partial state s is a **state** if $\mathrm{vars}(s) = \mathcal{V}$. I is an **initial state**. G is a partial state called **goal**, and a state s is a **goal state** iff $G \subseteq s$. Let p, t be partial states. We say that t **extends** p if $p \subseteq t$.

$\mathcal{O}$ is a finite set of **operators**, each operator $o \in \mathcal{O}$

has a precondition $\mathrm{pre}(o)$ and effect $\mathrm{eff}(o)$, which are partial states over $\mathcal{V}$, and a cost $\mathrm{c}(o) \in \mathbb{R}_0^+$. An operator o is **applicable** in a state s iff $\mathrm{pre}(o) \subseteq s$. The **resulting state** of applying an applicable operator o in a state s is another state $o[\![s]\!]$ such that $o[\![s]\!][V] = \mathrm{eff}(o)[V]$ for every $V \in \mathrm{vars}(\mathrm{eff}(o))$, and $o[\![s]\!][V] = s[V]$ for every $V \in \mathcal{V} \setminus \mathrm{vars}(\mathrm{eff}(o))$.

A sequence of operators $\pi = \langle o_1, \ldots, o_n \rangle$ is applicable in a state s_0 if there are states $s_1, \ldots, s_n$ such that o_i is applicable in s_{i-1} and $s_i = o_i[\![s_{i-1}]\!]$ for $i \in \{1, \ldots, n\}$. The resulting state of this application is $\pi[\![s_0]\!] = s_n$ and $\mathrm{c}(\pi) = \sum_{i=1}^{n} \mathrm{c}(o_i)$ denotes the cost of this sequence of operators. A sequence of operators π is called an **s-plan** iff π is applicable in a state s and $\pi[\![s]\!]$ is a goal state. An s-plan π is called **optimal** if its cost is minimal among all s-plans.

A state s is **reachable** if there exists an operator sequence π applicable in I such that $\pi[\![I]\!] = s$. Otherwise, we say that s is unreachable. The set of all reachable states is denoted by $\mathcal{R}$. An operator o is **reachable** iff it is applicable in some reachable state. A state s is a **dead-end state** iff $G \not\subseteq s$ and there is no s-plan.

A **heuristic** $h : \mathcal{R} \mapsto \mathbb{R} \cup \{\infty\}$ estimates the cost of optimal s-plans. The **optimal heuristic** $h^\star(s)$ maps each reachable state s to the cost of the optimal s-plan or to ∞ if s is a dead-end state. A heuristic h is called (a) **admissible** iff $h(s) \leq h^\star(s)$ for every reachable state $s \in \mathcal{R}$; (b) **goal-aware** iff $h(s) \leq 0$ for every reachable goal state s; and (c) **consistent** iff $h(s) \leq h(o[\![s]\!]) + \mathrm{c}(o)$ for all reachable states $s \in \mathcal{R}$ and operators $o \in \mathcal{O}$ applicable in s.

It is well-known that goal-aware and consistent heuristics are also admissible. Note that we define heuristics over the reachable states (instead of all states) because we intend to use heuristics in the (forward) heuristic search and because we want to use state invariants describing the reachable state space for an improvement of the heuristic values.

3 Mutexes and Disambiguation

Before we turn to the potential heuristic, we first introduce the notion of state invariants describing a mutual exclusion between facts and the notion of disambiguation of variables, which we later use for a generalization and improvement of potential heuristics.

Definition 1. Let Π denote a planning task with facts $\mathcal{F}$. A set of facts $M \subseteq \mathcal{F}$ is a **mutex** if $M \not\subseteq s$ for every reachable state $s \in \mathcal{R}$.

A mutex is a set of facts that never appear together in any reachable state. The most commonly used method for inference of mutexes is the h^m heuristic (Bonet and Geffner 2001; Alcázar and Torralba 2015) usually only for $m = 2$.

It is easy to see that every superset of a mutex is also a mutex and that every pair of facts from each variable form a mutex, because exactly one fact from each variable is present in every state. In this paper, we work with sets of mutexes. To simplify the notation we introduce the following notion of a mutex-set.

Definition 2. Let Π denote a planning task with variables $\mathcal{V}$ and facts $\mathcal{F}$. A set of sets of facts $\mathcal{M} \subseteq 2^{\mathcal{F}}$ is called a **mutex-set** if the following hold: (a) every $M \in \mathcal{M}$ is a mutex; and (b) for every $M \in \mathcal{M}$ and every $f \in \mathcal{F}$ it holds that $M \cup \{f\} \in \mathcal{M}$; and (c) for every variable $V \in \mathcal{V}$ and every pair of facts $f, f' \in \mathcal{F}_V$, $f \neq f'$, it holds that $\{f, f'\} \in \mathcal{M}$.

In other words, a mutex-set is an upper set of a set of mutexes and it always contains all mutexes that can be inferred directly from the variables of the FDR representation. This allows us to write, for example, $s \in \mathcal{M}$ for a state s and a mutex-set $\mathcal{M}$ when we want to say that s contains a subset of facts that is a mutex. But, of course, in practice we keep the inferred mutexes as a set without explicitly constructing all supersets.

When dealing with backward search (also known as regression), Alcázar et al. (2013) noted that mutexes can be used for extending partial states p with a value of some variable V not defined in p ($V \notin \mathrm{vars}(p)$), if all but one value is a mutex with p. Alcázar and Torralba (2015) re-used this idea for the pruning technique based on a fixpoint computation of h^2 heuristic in both forward and backward direction. They found out that this process is actually essential for the backward h^2 heuristic to be able to infer any useful information. We borrow and extend this notion, called disambiguation, in the following way.

Definition 3. Let Π denote a planning task with facts $\mathcal{F}$ and variables $\mathcal{V}$, let $V \in \mathcal{V}$ denote a variable, and let p denote a partial state. A set of facts $F \subseteq \mathcal{F}_V$ is called a **disambiguation of V for** p if for every reachable state $s \in \mathcal{R}$ such that $p \subseteq s$ it holds that $F \cap s \neq \emptyset$ (i.e., $\langle V, s[V] \rangle \in F$).

Clearly, every $\mathcal{F}_V$ is a disambiguation of V for all possible partial states. Moreover, it follows directly from the definition that if some $F \subseteq \mathcal{F}_V$ is a disambiguation of V for some partial state p then every $f \in \mathcal{F}_V \setminus F$ must be a mutex with p, i.e., every state s such that $\{f\} \cup p \subseteq s$ is unreachable. Therefore, if F is the empty set, then any state s such that $p \subseteq s$ is unreachable. So, as previously noted by Alcázar et al. (2013), we can use empty disambiguations to prune unreachable operators (if a precondition of an operator extends p), or to prove unsolvability of the planning task (if G extends p). Moreover, if the disambiguation of V consists of exactly one fact, then the partial state p can be safely extended with that fact, because it is the only value that can be assigned to the variable V in any reachable state $s \supseteq p$.

Given a partial state p and a mutex-set $\mathcal{M}$, we define a set $\mathcal{M}_p = \{f \mid f \in \mathcal{F}, p \cup \{f\} \in \mathcal{M}\}$ as the set of facts which $\mathcal{M}$ entails to be mutex with p. Note that $\mathcal{F}_V \setminus \mathcal{M}_p$ is a disambiguation of V for p.

Algorithm 1 encapsulates a use of disambiguation in a simple fixpoint algorithm. On line 4, the domain of each variable is reduced by removing facts that are mutex with the partial state p. On line 5, p is extended if there is only one possible fact that can appear in a reachable state containing p. On line 6, the algorithm reports p as a mutex if all facts from the variable's domain are mutex with p.

We extend the notion of disambiguation described by Alcázar et al. (2013) and Alcázar and Torralba (2015) in that we keep also the sets containing more than one fact and we show in the next section how these sets can be used for

Algorithm 1: Single-fact fixpoint disambiguation.

Input: A planning task Π with variables $\mathcal{V}$ and facts $\mathcal{F}$, a partial state p, and a mutex-set $\mathcal{M}$.
Output: A partial state p extended with disambiguations of size one.

1 **do**
2 $\quad p' \leftarrow p$;
3 $\quad$ **for each** $V \in \mathcal{V}$ **do**
4 $\quad\quad D_V \leftarrow \mathcal{F}_V \setminus \mathcal{M}_p$;
5 $\quad\quad$ **if** $|D_V| = 1$ **then** $p \leftarrow p \cup D_V$;
6 $\quad\quad$ **if** $|D_V| = 0$ **then return** "p is a mutex" ;
7 **while** $p' \neq p$;

strengthening potential heuristics.

Algorithm 2 shows an improved algorithm that computes disambiguations of all variables for a given partial state p. The algorithm keeps track of facts that cannot be part of a reachable state extending p (the set A). On line 7 the disambiguations are restricted by removing these facts, and on line 8 the set A is expanded by those facts that cannot be part of any reachable state extending any of $p \cup \{f\}$ for $f \in D_V$.

Theorem 4. *Algorithm 2 always produces a set of disambiguations of all variables for the given partial state p.*

Proof. The algorithm always terminates, because the sets D_V, for every $V \in \mathcal{V}$, can only decrease in size in each cycle (line 7), therefore there is a fixpoint.

Now we show that in every step, every D_V is a disambiguation of V for p. D_V is initialized with $\mathcal{F}_V$ (line 1) which is a disambiguation of V for p by definition. Then D_V is changed only on line 7 by removing the set A. So it suffices to show that at every point, A contains only the facts that cannot occur in any reachable state s extending p, i.e., $A \cap s = \emptyset$. This is true for the initialization of A (line 2). A is updated on line 8 by adding the set $X = \bigcap_{f \in D_V} \mathcal{M}_{p \cup \{f\}}$. Let $f' \in X$. By definition of X the fact f' cannot occur in any reachable state extending any of partial states $p \cup \{f\}$ for $f \in D_V$. Since D_V is a disambiguation of V for p, every reachable state s extending p extends $p \cup \{f\}$ for some $f \in D_V$. Therefore $s \cap X = \emptyset$. □

Note that the mutex-set $\mathcal{M}$ contains all mutex pairs from all variables so for every variable $V \in \mathrm{vars}(p)$ defined in the partial state p, D_V is set to $\{\langle V, p[V]\rangle\}$ in the first cycle D_V is changed on line 7. Also note that if one disambiguation is found out to be empty, and therefore p is proved to be mutex, then all disambiguations are gradually also set to empty sets.

4 Potential Heuristics

Potential heuristics, introduced by Pommerening et al. (2015a), assign a numerical value to each fact, and the heuristic value for a state s is then simply a sum of the potentials of all facts in s.

Definition 5. Let Π denote a planning task with facts $\mathcal{F}$. A **potential function** is a function $\mathtt{P} : \mathcal{F} \mapsto \mathbb{R}$. A **potential heuristic** for $\mathtt{P}$ maps each state $s \in \mathcal{R}$ to the sum of potentials of facts in s, i.e., $h^{\mathtt{P}}(s) = \sum_{f \in s} \mathtt{P}(f)$.

Algorithm 2: Multi-fact fixpoint disambiguation.

Input: A planning task Π with variables $\mathcal{V}$ and facts $\mathcal{F}$, a partial state p, and a mutex-set $\mathcal{M}$.
Output: A set of disambiguations $\mathfrak{D}$ of all variables $\mathcal{V}$ for p.

1 $D_V \leftarrow \mathcal{F}_V$ for every $V \in \mathcal{V}$;
2 $A \leftarrow \mathcal{M}_p$; `// A set of facts that are mutex with p`
3 **do**
4 $\quad$ change $\leftarrow$ False;
5 $\quad$ **for each** $V \in \mathcal{V}$ **do**
6 $\quad\quad$ **if** $D_V \setminus A \neq D_V$ **then**
7 $\quad\quad\quad D_V \leftarrow D_V \setminus A$;
8 $\quad\quad\quad A \leftarrow A \cup \bigcap_{f \in D_V} \mathcal{M}_{p \cup \{f\}}$;
9 $\quad\quad\quad$ change $\leftarrow$ True;
10 **while** change;
11 $\mathfrak{D} \leftarrow \{D_V \mid V \in \mathcal{V}\}$;

Pommerening et al. (2015a) described a set of inequalities that are sufficient conditions for the potential heuristic to be admissible, which can be formulated as the following theorem.[1]

Theorem 6. *Let $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G\rangle$ denote a planning task, $\mathtt{P}$ a potential function, and for every operator $o \in \mathcal{O}$, let $\mathrm{pre}^\star(o) = \{\langle V, \mathrm{pre}(o)[V]\rangle \mid V \in \mathrm{vars}(\mathrm{pre}(o)) \cap \mathrm{vars}(\mathrm{eff}(o))\}$ and $\mathrm{vars}^\star(o) = \mathrm{vars}(\mathrm{eff}(o)) \setminus \mathrm{vars}(\mathrm{pre}(o))$. If*

$$\sum_{f \in G} \mathtt{P}(f) + \sum_{V \in \mathcal{V} \setminus \mathrm{vars}(G)} \max_{f \in \mathcal{F}_V} \mathtt{P}(f) \leq 0 \qquad (1)$$

and for every operator $o \in \mathcal{O}$ it holds that

$$\sum_{f \in \mathrm{pre}^\star(o)} \mathtt{P}(f) + \sum_{V \in \mathrm{vars}^\star(o)} \max_{f \in \mathcal{F}_V} \mathtt{P}(f) - \sum_{f \in \mathrm{eff}(o)} \mathtt{P}(f) \leq \mathrm{c}(o), \quad (2)$$

then the potential heuristic for $\mathtt{P}$ is admissible.

Eq. (1) makes sure that the sum of potentials is goal-aware, and Eq. (2) ensures consistency of the potential heuristic. The theorem, however, does not tell us how to actually choose the potential function. Pommerening et al. (2015a) proposed to formulate the inequalities as constraints of a linear program (LP) and then a solution for any optimization function results in an admissible potential heuristic. The selection of optimization functions is discussed in the next section.

For now, move your attention to the maxima over all facts of the undefined variables in Eq. (1) and (2). For the goal equation Eq. (1), we do not know how the reachable goal states actually look like, so we prepare for the worst case by using the maximum potential over all facts of each undefined variable. Similarly for the operator equation Eq. (2), we do not fully know the reachable states where the operator is applicable. However, we have demonstrated in the previous section that mutexes can be used for narrowing down the unknown parts. So, we can use disambiguations to generalize Theorem 6 by the following theorem.

[1] The original formulation uses planning tasks in the so-called Transition Normal Form, but the general case is described in the technical report (Pommerening et al. 2015b).

Theorem 7. *Let $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$ denote a planning task with facts $\mathcal{F}$, and let $\mathtt{P}$ denote a potential function, and*

(i) for every variable $V \in \mathcal{V}$, let $G_V \subseteq \mathcal{F}_V$ denote a disambiguation of V for G s.t. $|G_V| \geq 1$, and

(ii) for every operator $o \in \mathcal{O}$ and every variable $V \in \mathrm{vars}(\mathrm{eff}(o))$, let $E^o_V \subseteq \mathcal{F}_V$ denote a disambiguation of V for $\mathrm{pre}(o)$ s.t. $|E^o_V| \geq 1$.

If

$$\sum_{V \in \mathcal{V}} \max_{f \in G_V} \mathtt{P}(f) \leq 0 \tag{3}$$

and for every operator $o \in \mathcal{O}$ it holds that

$$\sum_{V \in \mathrm{vars}(\mathrm{eff}(o))} \max_{f \in E^o_V} \mathtt{P}(f) - \sum_{f \in \mathrm{eff}(o)} \mathtt{P}(f) \leq \mathrm{c}(o), \tag{4}$$

then the potential heuristic for $\mathtt{P}$ is admissible.

Proof. To show that the potential heuristic is goal-aware, we need to prove that for every reachable goal state s_G it holds that $\sum_{f \in s_G} \mathtt{P}(f) \leq 0$. Let $f = \langle V, v \rangle \in s_G$. From (i), we have $f \in G_V$. Since for every G_V and every $f \in G_V$ it holds that $\mathtt{P}(f) \leq \max_{f' \in G_V} \mathtt{P}(f')$, then from Eq. (3) it follows that $\sum_{f \in s_G} \mathtt{P}(f) \leq \sum_{V \in \mathcal{V}} \max_{f \in G_V} \mathtt{P}(f) \leq 0$.

To show consistency, we need to prove that for every operator $o \in \mathcal{O}$ and every reachable state $s \in \mathcal{R}$ such that $\mathrm{pre}(o) \subseteq s$ it holds that $\sum_{f \in s} \mathtt{P}(f) - \sum_{f \in o[\![s]\!]} \mathtt{P}(f) \leq \mathrm{c}(o)$. Let $t = (s \cap o[\![s]\!]) \setminus \mathrm{eff}(o)$ be the part of s that is not affected by the operator o. Then clearly, $\sum_{f \in s} \mathtt{P}(f) - \sum_{f \in o[\![s]\!]} \mathtt{P}(f) = \sum_{f \in s \setminus t} \mathtt{P}(f) - \sum_{f \in o[\![s]\!] \setminus t} \mathtt{P}(f) = \sum_{f \in s \setminus t} \mathtt{P}(f) - \sum_{f \in \mathrm{eff}(o)} \mathtt{P}(f)$, because $o[\![s]\!] \setminus t = \mathrm{eff}(o)$. Furthermore, since $\mathrm{vars}(s \setminus t) = \mathrm{vars}(\mathrm{eff}(o))$, it follows that every $f = \langle V, v \rangle \in s \setminus t$ belongs to E^o_V and consequently $\mathtt{P}(f) \leq \max_{f' \in E^o_V} \mathtt{P}(f')$. Therefore (by Eq. (4)) $\sum_{f \in s \setminus t} \mathtt{P}(f) - \sum_{f \in \mathrm{eff}(o)} \mathtt{P}(f) \leq \sum_{V \in \mathrm{vars}(\mathrm{eff}(o))} \max_{f \in E^o_V} \mathtt{P}(f) - \sum_{f \in \mathrm{eff}(o)} \mathtt{P}(f) \leq \mathrm{c}(o)$. So $h^{\mathtt{P}}$ is goal-aware and consistent and therefore admissible. □

Note that the requirement on the non-empty disambiguations in (i) and (ii) is just to simplify the theorem, because for every empty disambiguation we could either remove the corresponding operator (if E^o_V is empty), or report the planning task unsolvable (if G_V is empty), as we already explained.

Clearly, Eq. (3) generalizes Eq. (1) because for every $V \in \mathrm{vars}(G)$ the singleton $\{\langle V, G[V] \rangle\}$ is a disambiguation of V for G, and for every $V \in \mathcal{V} \setminus \mathrm{vars}(G)$ the set $\mathcal{F}_V$ is a disambiguation of V for G. Thus Theorem 7 allows to use proper subsets of $\mathcal{F}_V$ for the variables undefined in G. Similarly, Eq. (4) generalizes Eq. (2) because we can use the same reasoning for the preconditions of operators.

4.1 Transition Normal Form

Originally, potential heuristics were formulated for planning tasks in the so-called Transition Normal Form (TNF) (Pommerening and Helmert 2015). A planning task is in TNF if the goal is fully defined ($\mathrm{vars}(G) = \mathcal{V}$) and $\mathrm{vars}(\mathrm{pre}(o)) = \mathrm{vars}(\mathrm{eff}(o))$ for every operator o. Any planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$ can be compiled into TNF as follows:

- Add a fresh value U to the domain of every variable.
- For every variable $V \in \mathcal{V}$ and every fact $f \in \mathcal{F}_V$, $f \neq \langle V, U \rangle$, add a new *forgetting* operator o_f with $\mathrm{pre}(o_f) = \{f\}$ and $\mathrm{eff}(o_f) = \{\langle V, U \rangle\}$ and the cost $\mathrm{c}(o_f) = 0$.
- For every operator $o \in \mathcal{O}$ and every variable $V \in \mathcal{V}$:
  - If $V \in \mathrm{vars}(\mathrm{pre}(o))$ and $V \notin \mathrm{vars}(\mathrm{eff}(o))$, then add $\langle V, \mathrm{pre}(o)[V] \rangle$ to $\mathrm{eff}(o)$.
  - If $V \in \mathrm{vars}(\mathrm{eff}(o))$ and $V \notin \mathrm{vars}(\mathrm{pre}(o))$, then add $\langle V, U \rangle$ to $\mathrm{pre}(o)$.
- For every $V \in \mathcal{V} \setminus \mathrm{vars}(G)$ add $\langle V, U \rangle$ to G.

There is a clear correspondence between the compilation into TNF and the LP formulation of Eq. (1) and (2) from Theorem 6 (Pommerening and Helmert 2015). When translating inequalities Eq. (1) and (2) to the LP constraints, we (a) create the LP variable X_f for every fact $f \in \mathcal{F}$ (holding the potential $\mathtt{P}(f)$), and (b) to deal with the maxima, we create another auxiliary LP variable M_V for every variable $V \in \mathcal{V}$ and add the constraint $X_f \leq M_V$ for every $f \in \mathcal{F}_V$. Then we can rewrite Eq. (1) to the constraint

$$\sum_{f \in G} X_f + \sum_{V \in \mathcal{V} \setminus \mathrm{vars}(G)} M_V \leq 0, \tag{5}$$

and similarly Eq. (2) to constraints

$$\sum_{f \in \mathrm{pre}^\star(o)} X_f + \sum_{V \in \mathrm{vars}^\star(o)} M_V - \sum_{f \in \mathrm{eff}(o)} X_f \leq \mathrm{c}(o), \tag{6}$$

for every operator $o \in \mathcal{O}$, and look for the maximization over some optimization function. Compare Eq. (5) and Eq. (6) to the constraints resulting from the planning task compiled into TNF and you will find that we get exactly the same constraints where LP variables M_V correspond to the special value U added to every variable in TNF.

The generalization of Theorem 6 via disambiguations transposes also to the TNF. Instead of creating a single fresh value U for every variable, we create fresh values U_{G_V} and $U_{E^o_V}$ for every disambiguation G_V and E^o_V, respectively. Then we use these values in the same way U values are used in the original TNF formulation. There will be forgetting operators for every U_{G_V} and $U_{E^o_V}$ that go over the facts in their respective disambiguations rather than over all values from the corresponding domain. Instead of adding $\langle V, U \rangle$ into operators' preconditions, we add $\langle V, U_{E^o_V} \rangle$. And instead of adding $\langle V, U \rangle$ into the goal, we add $\langle V, U_{G_V} \rangle$.

Pommerening and Helmert (2015) showed that the compilation to TNF can produce a planning task twice the size of the original one, in the worst case. With disambiguations, the compilation to TNF can grow even more, but it is still polynomially bounded. Although we choose disambiguations from the powerset $2^{\mathcal{F}}$, the actual number of disambiguations is limited by the number of operators and the size of their preconditions. So, the number of U_{G_V} values can be at most $|\mathcal{V}|$, and the number of $U_{E^o_V}$ cannot be more than $|\mathcal{O}| \cdot |\mathcal{V}|$. The maximum number of forgetting operators then corresponds to these limits.

However, the resulting representation can also be smaller than the original TNF, because the disambiguations of size

one can get rid of U values completely, and the disambiguations that are proper subsets of the corresponding $\mathcal{F}_V$ produce fewer forgetting operators. In fact, the experimental evaluation on the domains from International Planning Competitions (IPCs) shows that the representation with disambiguations is never bigger than without disambiguations.

4.2 Extension to Dead-End States

As we already mentioned, Alcázar and Torralba (2015) proposed an algorithm for pruning of planning tasks by computing the h^2 heuristic in both forward and backward direction. This algorithm finds so called *dead pairs* (Eriksson, Röger, and Helmert 2018), i.e., pairs of facts that are either mutexes (as per Definition 1) or *backward mutexes* found in the backward direction. Dead pairs correspond to both unreachable and dead-end states, i.e., given a dead pair p, every state s such that $p \subseteq s$ is either unreachable or it is a dead-end state.

It is possible to use dead pairs instead of mutexes in the computations of disambiguations. A disambiguation for p computed with dead pairs contains facts that appear in reachable states extending p that are not dead-end states, i.e., this type of disambiguation differs from Definition 3, because it excludes reachable states that are dead-end states. However, this type of disambiguation can be used as G_V and E^o_V in Theorem 7 and the potential heuristic remains admissible. This follows from the fact that the optimal heuristic value for dead-end states is ∞. Hence any heuristic value is admissible for them. In this paper, we investigate only disambiguations found with mutexes, but the extension with dead pairs should be straightforward.

5 Optimization Functions

When Pommerening et al. (2015a) introduced potential heuristics, they used the optimization function for the initial state

$$\mathrm{opt}_I = \sum_{f \in I} \mathtt{P}(f). \tag{7}$$

Maximization of opt_I subject to the constraints from Theorem 7 (or Theorem 6) yields the highest possible heuristic value (h-value) for the initial state. However, maximization of opt_I does not provide an incentive for optimizing potentials of the facts that do not appear in the initial state (at least not directly). Of course, one could recompute potentials for each state reached during the search. That would always provide the best possible h-value but it would also be too costly from the computational point of view.

Seipp, Pommerening, and Helmert (2015) studied different optimization functions. One of their main contributions is the "automatic diversification" algorithm for finding an ensemble of potential heuristics constructed from a set of states sampled by random walks. We will experimentally evaluate the effect of disambiguations on this variant, but, for space reasons, we refer readers interested in the detailed description of automatic diversification to the original paper. Another contribution was the introduction of a family of optimization functions aiming at maximizing the average h-value over all reachable states. And this is the obvious place where mutexes can be utilized.

5.1 All States Potentials

The perfect optimization function of which maximization yields the maximum average h-value over all reachable states is the weighted sum of the potentials over all reachable states:

$$\mathrm{opt}_{\mathcal{R}} = \frac{1}{|\mathcal{R}|} \sum_{s \in \mathcal{R}} \sum_{f \in s} \mathtt{P}(f). \tag{8}$$

By reordering summands in Eq. (8), $\mathrm{opt}_{\mathcal{R}}$ can be written as $\sum_{f \in \mathcal{F}} \alpha_f \mathtt{P}(f)$ where the coefficient α_f is just the probability that a randomly chosen reachable state contains f. Consequently, by maximizing $\mathrm{opt}_{\mathcal{R}}$ we look for potentials such that the corresponding potential heuristic maximizes its expected value.

Listing all reachable states is, obviously, infeasible, and so is uniform sampling of reachable states. So, as an approximation, Seipp, Pommerening, and Helmert (2015) proposed to adopt the approach of Haslum et al. (2007) and sample the states $S \subseteq \mathcal{R}$ by random walks starting from the initial state with a binomially distributed length of the walks centered around the double of the maximum h-value for the initial state, leading to the following optimization function:

$$\mathrm{opt}_{\hat{S}} = \frac{1}{|S|} \sum_{s \in S} \sum_{f \in s} \mathtt{P}(f). \tag{9}$$

Another proposed option was to count all syntactic states (i.e., all possible assignments to variables):

$$\mathrm{opt}_{\mathcal{S}} = \sum_{\langle V, v \rangle \in \mathcal{F}} \frac{1}{|\mathrm{dom}(V)|} \mathtt{P}(\langle V, v \rangle). \tag{10}$$

This approach assumes the uniform distribution of the values within their respective domains over all reachable states which is $1/\left|\mathrm{dom}(V)\right|$ for every fact $\langle V, v \rangle$.

We extend the idea of using all syntactic states by taking mutexes into account. Suppose we want to estimate the number of states containing a fact $f = \langle V, v \rangle \in \mathcal{F}$. The simplest estimate of the number of these (syntactic) states is to compute a product of sizes of all variables' domains except V, because the variable V is assumed to be already set to v: $\prod_{V' \in \mathcal{V} \setminus \{V\}} |\mathrm{dom}(V')|$. This estimate is actually an upper bound on the true number of reachable states containing f.

However, if we take mutexes into account, we could remove the facts that are mutex with f from all domains and compute the product of sizes of these reduced domains. The resulting estimate would necessarily be lower than (or equal to) the previous one. It would be an upper bound too, because we are removing only the facts that are certainly not part of the reachable states containing f. Therefore, we certainly get a better estimate. Moreover, we can extend this idea to partial states, i.e., instead of asking how many reachable states contain a single fact f, we can ask how many reachable states extend a partial state p.

For a given number $1 \leq k \leq |\mathcal{V}|$ and a partial state t we define a set $\mathcal{P}^t_k$ as the set of all partial states of size k extending t. Further recall that for a mutex-set $\mathcal{M}$ and a partial state p we defined the set $\mathcal{M}_p = \{f \mid f \in \mathcal{F}, p \cup \{f\} \in \mathcal{M}\}$.

Now we can estimate the number of reachable states extending a partial state p using a given mutex-set $\mathcal{M}$ by the product $\prod_{V \in \mathcal{V}} |\mathcal{F}_V \setminus \mathcal{M}_p|$. Note that if p is a mutex ($p \in \mathcal{M}$), then $\mathcal{M}_p$ contains all facts and the product is zero. Also, since $\mathcal{M}$ always contains mutex pairs from all variables, then $|\mathcal{F}_V \setminus \mathcal{M}_p| = 1$ for all $V \in \mathrm{vars}(p)$ if $p \notin \mathcal{M}$.

With all building blocks in place, we can define

$$\mathcal{C}^k_f(\mathcal{M}) = \sum_{p \in \mathcal{P}^{\{f\}}_k} \prod_{V \in \mathcal{V}} |\mathcal{F}_V \setminus \mathcal{M}_p| \tag{11}$$

as an estimation of the number of reachable states containing the fact f while considering mutex-set $\mathcal{M}$ and all partial states of size k. The corresponding optimization function is:

$$\mathrm{opt}^k_{\mathcal{M}} = \sum_{f=\langle V,v\rangle \in \mathcal{F}} \frac{\mathcal{C}^k_f(\mathcal{M})}{\sum_{f' \in \mathcal{F}_V} \mathcal{C}^k_{f'}(\mathcal{M})} \mathtt{P}(f). \tag{12}$$

That is, we choose a number $k \geq 1$ and for every fact f and every partial state p of size k containing f, we compute the estimation of the number of reachable states extending p (the inner product in Eq. (11)). To get the estimate for a single fact f, we sum over the estimates for partial states p containing f (this is the number $\mathcal{C}^k_f(\mathcal{M})$). Finally, for every variable $V \in \mathcal{V}$ we normalize the collection of $\mathcal{C}^k_f(\mathcal{M})$ for $f \in \mathcal{F}_V$ so that it forms a probability distribution estimating the actual probability that a fact $f \in \mathcal{F}_V$ appears in a randomly chosen reachable state. In other words, instead of using the uniform distribution as in Eq. (10), i.e., $1/\left|\mathrm{dom}(V)\right|$, we use mutexes and estimate the number of states step-by-step for all partial states of size k and then sum these counts to the final estimation.

5.2 Conditioned Ensemble of All States Potentials

Averaging sampled states $S \subseteq \mathcal{R}$, as in $\mathrm{opt}_{\hat{S}}$, is not the only way S can be used for a construction of a potential heuristic. Seipp, Pommerening, and Helmert (2015) proposed to use an ensemble of potential heuristics, for example one potential heuristic per state from S, and then use the maximum h-value from all potential heuristic as the h-value for the given state.

The optimization for all reachable states $\mathrm{opt}_{\mathcal{R}}$ tackles the problem of finding the best potentials by maximizing the expected value of the sum of potentials for a randomly chosen reachable state. However, if we somehow divide the whole reachable state space into sets of states $S_1 \cup \ldots \cup S_n = \mathcal{R}$, then averaging over S_i with $\mathrm{opt}_{\hat{S}_i}$ would give us better h-values (at average) for each set S_i. Then we could use the ensemble of potential heuristics optimized for $\mathrm{opt}_{\hat{S}_i}$ for all $i = \{1, \ldots, n\}$ and the maximum of h-values for a given state over all these heuristics should give us a better resulting h-value. Intuitively, we can see this approach as being halfway between $\mathrm{opt}_{\mathcal{R}}$, and computing potentials for each individual state. Unfortunately, we do not know how to select the sets S_i or how to sample them efficiently. We can, however, re-use the approach to $\mathrm{opt}^k_{\mathcal{M}}$ in constructing a similar ensemble.

Taking the idea of using mutexes one step further, we can optimize for the maximum average potentials over the states extending a partial state t. In other words, we can fix the partial state t as a sort of selector for states and then use the same idea as for $\mathrm{opt}^k_{\mathcal{M}}$ except that we count only the states extending t. We start with a slight modification of Eq. (11) by restricting the count to a given partial state t:

$$\mathcal{K}^k_f(\mathcal{M}, t) = \sum_{p \in \mathcal{P}^{t \cup \{f\}}_{|t|+k}} \prod_{V \in \mathcal{V}} |\mathcal{F}_V \setminus \mathcal{M}_p| \,. \tag{13}$$

In words, $\mathcal{K}^k_f$ differs from $\mathcal{C}^k_f$ in that $\mathcal{K}^k_f$ takes the partial states p of size $|t|+k$ extending $t \cup \{f\}$ instead of the partial states of size k containing f. We also tacitly assume that $t \cup \{f\}$ is a partial state, i.e., either $f \in t$ or the variable V corresponding to the fact f does not belong to $\mathrm{vars}(t)$. For the cases where $V \in \mathrm{vars}(t)$, we define $\mathcal{K}^k_f(\mathcal{M}, t) = 0$.

The optimization function for a fixed partial state t is a small modification of Eq. (12) where we replace $\mathcal{C}^k_f$ with $\mathcal{K}^k_f$:

$$\mathrm{opt}^{t,k}_{\mathcal{M}} = \sum_{f=\langle V,v\rangle \in \mathcal{F}} \frac{\mathcal{K}^k_f(\mathcal{M}, t)}{\sum_{f' \in \mathcal{F}_V} \mathcal{K}^k_{f'}(\mathcal{M}, t)} \mathtt{P}(f). \tag{14}$$

The remaining question is how to select the partial states t on which to condition potential heuristics in the ensemble. In this paper, we evaluate only uniformly randomly sampled partial states of size 1 and 2. However, the question left for future research is whether we can use some sort of structural information, such as causal graphs or some kind of a relation between mutexes, for the selection of the best possible sets.

5.3 Adding Constraint on Initial State

As already mentioned, the disadvantage of optimizing for the initial state (opt_I) is that the optimization function does not provide an incentive to optimize the potentials that are not part of the initial state. But it should provide good heuristic values in the vicinity of the initial state, assuming the operators change only few facts at the time. Conversely, the optimization for all states, in all variants, including $\mathrm{opt}_{\mathcal{R}}$, does not target any particular state and the resulting h-values highly depend on what the reachable state space actually looks like. For example, a huge number of goal states can unintentionally decrease the average h-values even though the goal-awareness of potential heuristics is explicitly enforced by a constraint and we actually want to push the h-values higher for all states but the goal states.

We can, however, overcome this behaviour at least partially by combining the optimization for the initial state and for all states. Let $h^{\mathtt{P}}_I$ denote the potential heuristic optimized for the initial state. Then using an optimization for all states (any discussed variant) and imposing the additional constraint

$$\sum_{f \in I} \mathtt{P}(f) = h^{\mathtt{P}}_I(I) \tag{15}$$

will force the optimizer to find potentials that will produce a high h-value for the initial state while maximizing average h-values for states containing facts that are not part of the initial state. So, during the search, we should get more accurate h-values from the beginning of the search and as we

get farther from the initial state and closer to the goal, the potentials optimized for the average case should take over.

This, of course, requires to compute the potentials twice. The first time for the h-value $h^{\mathtt{P}}_I(I)$. And the second time for the all states potentials using Eq. (15).

6 Experimental Evaluation

The evaluated methods were all implemented[2] in C and experimentally evaluated with the Fast Downward planner (Helmert 2006) on a cluster of computing nodes with Intel Xeon Scalable Gold 6146 processors and CPLEX solver v12.9. The time and memory limits were set to 30 minutes and 8 GB, respectively. Operators and facts are pruned with the h^2 heuristic in forward and backward direction (Alcázar and Torralba 2015), and the translation from PDDL to FDR uses the inference of mutex groups proposed by Fišer (2020). However, for the methods that require mutexes, we computed them again with the h^2 heuristic because we can use only the forward direction for mutexes as per Definition 1 and we wanted to account for the increased computational demand for these methods. We used all planning domains from International Planning Competitions (IPCs) from 1998 to 2018 excluding the ones containing conditional effects after translation (leaving 65 domains).

We refer to the compared variants of potential heuristics as follows:

- $\mathtt{N}$: the variant without disambiguation,
- $\mathtt{D}$: the multi-fact disambiguation (Algorithm 2),
- $\mathtt{D}_1$: the single-fact disambiguation (Algorithm 1),
- $\mathtt{Init}$: the optimization for the initial state (opt_I),
- $\mathtt{All}$: the optimization for all syntactic states ($\mathrm{opt}_{\mathcal{S}}$),
- $\mathtt{MaxIA}$: the maximization over $\mathtt{Init}$ and $\mathtt{All}$,
- $\mathtt{Div}_n$: the diversification algorithm with n samples,
- $\hat{\mathtt{S}}_n$: the optimization for n randomly sampled states ($\mathrm{opt}_{\hat{\mathcal{S}}}$),
- $\mathtt{S}_n$: the maximization over n potential heuristics each optimized for a randomly sampled state,
- $\mathtt{M}^k$: the optimization for all states considering inferred mutexes ($\mathrm{opt}^k_{\mathcal{M}}$),
- $\mathtt{K}^k_n$: the maximization over n potential heuristics each optimized for all states considering mutexes and conditioned on a randomly sampled fact f ($\mathrm{opt}^{\{f\},k}_{\mathcal{M}}$).
- $\mathtt{L}^k_n$: the same as $\mathtt{K}^k_n$ except partial states $\{f_1, f_2\}$ of size two are sampled ($\mathrm{opt}^{\{f_1,f_2\},k}_{\mathcal{M}}$).

If the additional constraint on the initial state was used (Section 5.3, Eq. (15)), we append $\mathtt{+I}$. We fixed seeds for random number generators in order to get comparable results for the variants that use sampling. $\mathtt{Div}_n$ and $\hat{\mathtt{S}}_n$ are evaluated for $n = 1000$ to compare them to the results of Seipp, Pommerening, and Helmert (2015).

In Fig. 1, we show scatter plots of the heuristic values for the initial state when optimized for the initial state with and without disambiguation. The plots clearly show the advantage of utilizing multi-fact disambiguation in comparison to both single-fact disambiguation (right) and no disambiguation (left).

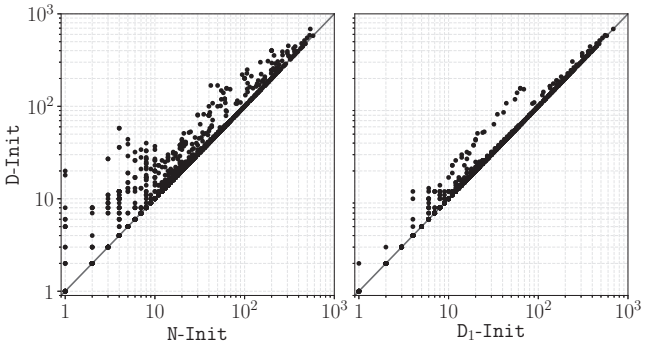

Figure 1: Heuristic values for the initial state with the zero h-values filtered out and the h-values in the parcprinter domain scaled by 10^{-4} to keep the plots compact.

We tried to fit as many results comparing the use of disambiguations as possible into Table 1 to show the positive effect of disambiguations on the number of solved tasks. (We added the row with the sum without the freecell domain because this domain contains considerably more planning tasks than other domains which may skew the overall results.) Disambiguations decrease the coverage only in a few domains, but never overall. The most significant decrease can be found in the blocks domain for $\mathtt{Init}$ and freecell00 for $\mathtt{All}$, but otherwise the decrease is just one or two fewer problem solved. Overall, the increase in coverage due to the (multi-fact) disambiguation ranges from 1.6% for $\mathtt{M}^2$ to 3.8% for $\mathtt{All+I}$ and it is two or more percent for all methods proposed by Seipp, Pommerening, and Helmert (2015) (similar results are obtained also if freecell00 is not counted).

The only difference between $\mathtt{M}^1$ and $\mathtt{M}^2$ is in the parcprinter domains (in favor of $\mathtt{M}^1$) which suggests that increasing k increases the computational intensity but does not provide much more accurate estimates of the number of reachable states. Overall, both $\mathtt{M}^1$ and $\mathtt{M}^2$ have higher coverage than $\mathtt{All}$ (with or without disambiguations), but most of the difference is due to the freecell domain. If the freecell domain is filtered out, $\mathtt{All}$ solves more tasks. The average from a sample of 1000 states ($\hat{\mathtt{S}}_{1\mathrm{k}}$) also seems to provide better results (both overall and in per-domain comparison).

However, constructing ensembles of 100 or 50 potential heuristics with $\mathtt{K}^1_{100}$ and $\mathtt{K}^1_{50}$ results in a higher coverage than $\mathtt{All}$, $\mathtt{M}^1$, $\mathtt{M}^2$, and $\hat{\mathtt{S}}_{1\mathrm{k}}$, which all optimize for the average state. The coverage is also higher than $\mathtt{Div}_{1\mathrm{k}}$ and $\mathtt{S}_{100}$ that maximize over an ensemble of potential heuristics as $\mathtt{K}^k_n$ does. We should note here that $\mathtt{K}^k_n$ conceptually sits between optimization for the average state and maximization over a sample of states, because $\mathtt{K}^k_n$ tries to do a little bit of both.

Table 2 compares the sampling based methods (with disambiguations) for different numbers of samples. The more samples are used in the ensemble the longer it takes to evaluate a state during the search and therefore the advantage of using potential heuristics diminishes. The sweet spot for

[2] https://gitlab.com/danfis/cpddl, branch icaps20-potentials

domain	Init			All			M^1		M^2		Div_{1k}		$\hat{S}_{1k}$		S_{100}	K^1_{100}	K^1_{50}	MaxIA		All+I		$\hat{S}_{1k}$+I		M^1+I
	N	D_1	D	N	D_1	D	N	D	N	D	N	D	N	D	D	D	D	N	D	N	D	N	D	D
agricola18 (20)	1	1	**3**	1	1	**3**	1	1	1	1	1	**3**	1	**3**	**3**	**3**	**3**	1	**3**	1	**3**	1	**3**	∘**3**
airport04 (50)	29	32	32	30	31	31	30	31	30	31	30	31	30	28	30	29	30	30	33	+30	⊕**36**	+30	⊕33	∘32
blocks00 (35)	**28**	21	21	**28**	21	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	21	**28**	**28**	**28**	**28**	**28**	+**28**	**28**	+**28**	+**28**
depot02 (22)	7	9	8	11	11	11	11	11	11	11	10	11	11	11	10	**12**	**12**	11	11	+11	+11	+11	+11	+11
freecell00 (80)	67	71	71	41	36	36	58	58	58	58	**73**	**73**	**73**	72	71	64	65	67	71	⊕72	⊕72	+**73**	⊕**73**	∘69
ged14 (20)	15	15	15	15	15	15	15	15	15	15	17	**19**	**19**	**19**	**19**	**19**	**19**	15	15	15	15	+**19**	+**19**	15
hiking14 (20)	13	13	13	**14**	**14**	**14**	**14**	**14**	**14**	**14**	11	11	**14**	**14**	**14**	13	13	**14**	**14**	+**14**	+**14**	+**14**	+**14**	+**14**
logistics00 (28)	11	11	11	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	16	**19**	**19**	**19**	**19**	+**19**	+**19**	+**19**	+**19**	+**19**
logistics98 (35)	2	2	2	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	3	**5**	**5**	**5**	**5**	+**5**	+**5**	+**5**	+**5**	+**5**
mystery98 (30)	17	17	**18**	17	17	**18**	17	**18**	17	**18**	17	**18**	17	**18**	**18**	**18**	**18**	17	**18**	17	**18**	17	**18**	**18**
nomystery11 (20)	10	10	10	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	10	**14**	**14**	**14**	**14**	+**14**	+**14**	+**14**	+**14**	+**14**
openstacks06 (30)	7	**14**	13	7	**14**	**14**	7	7	7	7	7	7	7	7	7	10	10	7	**14**	7	−13	7	∘13	∘13
openstacks08 (30)	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	21	**23**	**23**	**23**	**23**	**23**	**23**	**23**	−21	**23**	**23**	**23**
openstacks11 (20)	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	16	**18**	**18**	**18**	**18**	**18**	**18**	**18**	−16	**18**	**18**	**18**
openstacks14 (20)	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	0	**3**	**3**	2	2	**3**	**3**	**3**	**3**	−0	**3**	**3**
parcprinter08 (30)	27	27	27	21	20	20	24	25	23	23	17	17	17	16	16	**28**	**28**	**28**	27	∘25	⊕**28**	∘22	∘22	−25
parcprinter11 (20)	**20**	**20**	**20**	16	15	15	17	18	17	17	13	13	13	12	12	**20**	**20**	**20**	**20**	∘**20**	∘**20**	∘17	∘17	−17
parking11 (20)	7	6	6	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	+**8**	+**8**	+**8**	+**8**	+**8**
parking14 (20)	7	6	6	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	+**8**	+**8**	+**8**	+**8**	+**8**
pegsol08 (30)	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	**30**	**30**	29	29	29	29	29	29	29
pegsol11 (20)	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	**20**	**20**	19	19	19	19	19	19	19
petr-net-align18 (20)	**13**	**13**	**13**	7	9	9	7	7	7	7	11	11	11	11	9	7	11	**13**	**13**	∘12	∘**13**	−11	−11	∘**13**
pipesworld-notank04 (50)	26	26	25	21	25	25	20	24	20	24	25	26	27	**30**	27	24	25	26	28	∘26	⊕**30**	+27	+29	⊕29
pipesworld-tank04 (50)	18	17	17	16	17	16	17	17	17	17	14	18	19	19	**21**	20	20	18	17	⊕19	⊕19	⊕20	⊕20	⊕19
rovers06 (40)	6	6	6	7	**8**	**8**	7	**8**	7	**8**	7	**8**	7	**8**	6	**8**	**8**	7	**8**	+7	+**8**	+7	+**8**	+**8**
scanalyzer08 (30)	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**	12	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**
scanalyzer11 (20)	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	9	**10**	**10**	9	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
snake18 (20)	13	15	15	14	14	14	12	12	12	12	10	10	13	**17**	15	11	12	14	15	⊕15	∘15	13	−15	∘15
sokoban08 (30)	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	29	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
spider18 (20)	14	13	15	12	14	14	13	13	13	13	12	14	15	**16**	14	13	13	15	15	⊕**16**	∘15	+15	−15	∘15
storage06 (30)	**16**	**16**	15	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	+**16**	+**16**	**16**	+**16**	+**16**
termes18 (20)	12	12	12	**13**	**13**	**13**	**13**	**13**	**13**	**13**	12	12	12	12	12	**13**	**13**	**13**	**13**	−12	−12	12	12	−12
tetris14 (17)	15	16	15	14	14	15	16	16	16	16	**17**	**17**	**17**	**17**	**17**	**17**	**17**	**17**	**17**	⊕**17**	+**17**	+**17**	+**17**	⊕**17**
tidybot11 (20)	14	16	**18**	14	17	**18**	14	**18**	14	**18**	14	**18**	14	**18**	**18**	**18**	**18**	14	**18**	14	**18**	14	**18**	**18**
tidybot14 (20)	10	12	**14**	10	13	**14**	10	**14**	10	**14**	10	**14**	10	**14**	**14**	**14**	**14**	10	**14**	10	**14**	10	**14**	**14**
tpp06 (30)	6	6	6	6	6	6	6	6	6	6	**8**	**8**	7	7	6	6	6	7	7	⊕**8**	⊕**8**	⊕**8**	⊕**8**	⊕**8**
trucks06 (30)	13	13	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	+**14**	**14**	+**14**	**14**	**14**
visitall11 (20)	16	16	16	**17**	**17**	**17**	**17**	**17**	**17**	**17**	16	16	14	14	11	**17**	**17**	**17**	**17**	+**17**	+**17**	−14	−14	+**17**
visitall14 (20)	12	12	12	13	13	13	13	13	13	13	11	11	9	9	5	**14**	**14**	13	13	+13	+13	−9	−9	+13
woodworking08 (30)	12	12	12	14	14	15	12	12	12	12	16	**17**	15	16	15	12	12	14	15	+14	⊕**17**	+15	⊕**17**	⊕**17**
woodworking11 (20)	7	7	7	9	9	10	7	7	7	7	11	**12**	10	11	10	7	7	9	10	+9	⊕**12**	+10	⊕**12**	⊕**12**
zenotravel02 (20)	10	10	10	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	10	10	**11**	**11**	**11**	**11**	**11**	+**11**	+**11**	⊕**11**	⊕**11**	+**11**
Σ (1697)	921	933	938	903	913	927	921	938	920	935	932	957	937	961	925	961	969	960	989	964	**1 001**	949	985	987
Σ w/o freecell00 (1617)	854	862	867	862	877	891	863	880	862	877	859	884	864	889	854	897	904	893	918	892	**929**	876	912	918

Table 1: Number of solved tasks for different heuristics. Only the domains with a difference in coverage are listed.

n	5	10	50	100	250	500	750	1 000	2 000
D-S_n	896	910	923	**925**	897	863	828	791	741
D-K^1_n	951	957	**969**	961	947	935	925	925	899
D-K^2_n	953	960	**967**	956	944	925	917	919	912
D-L^1_n	934	951	**965**	963	932	877	850	824	765
D-L^2_n	937	951	**962**	957	928	874	844	810	744

Table 2: Number of solved tasks for a different number of samples n.

S_n seems to be at 100 samples and it is at 50 samples for the methods using mutexes. It also seems that increasing k (D-K^1_n vs. D-K^2_n, and D-L^1_n vs. D-L^2_n) does not provide better h-values, and the same holds for increasing the size of sampled partial states (D-K^1_n vs. D-L^1_n and D-L^2_n).

The best results were achieved with adding the constraint on the heuristic value of the initial state (Section 5.3, Eq. (15)) in combination with disambiguations. In Table 1, for the "+I" columns, we added (a) "∘" to indicate that the coverage in the respective domain is higher than for the corresponding variant without the constraint (without +I); (b) "+" to indicate a higher coverage than the Init variant; (c) "⊕" if both of the previous cases hold at the same time; and (d) "−" to indicate that the coverage is smaller than either of the variants. There are very few domains in which the coverage is smaller than for the variant without the added constraint or for Init (the cases with "−"): only 2 out of 65 for D-All+I, 4 for D-$\hat{S}_{1k}$+I, and only 3 for D-M^1+I. So, it indeed seems that the combination of the optimization for all states and for the initial state at the same time brings the best from both.

A similar approach to All+I is to compute two potential functions, for all states and for the initial state, separately and take the maximum as a heuristic value (denoted by MaxIA). The comparison between All+I and MaxIA shows that incorporating the initial state as a constraint (All+I) provides better results overall, but in some domains MaxIA solves more tasks than All+I. In parcprinter08, petri-net-alignment18, and termes18, N-MaxIA solves more tasks than N-All+I, and in openstacks06 D-MaxIA solves more tasks than D-All+I. In all other domains, All+I is at least as good as MaxIA.

To compare the results to other state-of-the-art heuristics, we also evaluated the LM-Cut (lmc) heuristic (Helmert and Domshlak 2009), the merge-and-shrink (ms) heuristic with SCC-DFP merge strategy and non-greedy bisimulation shrink strategy (Helmert et al. 2014; Sievers, Wehrle, and

	lmc	ms	flw	comp1	comp2	ppdbs	scrp
overall	911	895	816	1 046	1 091	1 090	1 112
w/o freecell00	896	874	762	1 019	1 059	1 056	1 040

Table 3: Number of solved tasks for a different heuristic search planners.

Helmert 2016), the flow (flw) heuristic (Bonet and van den Briel 2014; Bonet 2013), and the four best-performing non-portfolio planners from IPC 2018: Complementary1 (comp1) (Franco et al. 2018), and Complementary2 (comp2) (Franco et al. 2017; Franco, Lelis, and Barley 2018) planners, the Planning-PDBs planner (ppdbs) (Moraru et al. 2018; Franco et al. 2017), and the Scorpion planner (scrp) (Seipp 2018; Seipp and Helmert 2018). The overall coverage is shown in Table 3.

Our best variant of the potential heuristic (D-All+I) solved 90 (33 without the freecell domain) more tasks than lmc, 106 (55) more than ms, and 185 (167) more than flw. And the best performing variant that uses mutexes for estimating the number of reachable states, D-K^1_{50}, solved 58 (8), 74 (30), and 153 (142) more tasks than lmc, ms, and flw, respectively. However, all variants of the potential heuristic were surpassed by all four evaluated planners from IPC 2018. D-All+I solved 45 (90), 90 (130), 89 (127), and 111 (111) fewer planning tasks than comp1, comp2, ppdbs, and scrp, respectively.

7 Conclusion

We showed that utilizing mutexes can significantly improve potential heuristics. Disambiguations in the goal and operator preconditions increase the overall coverage whenever they are used and they decrease the coverage only in a very small number of domains. It is also clear that the multi-fact disambiguation dominates the single-fact disambiguation for virtually no cost at all.

A more accurate estimation of the number of reachable states using mutexes results in a higher overall coverage than the optimization for all syntactic states, but mainly due to two domains only. However, a similar technique used for an ensemble of potential heuristics, each conditioned on a partial state, increases the number of solved tasks even if the partial states are sampled randomly. A better structural analysis, for example with causal graphs, could probably lead to even better selection of the partial states, but we leave it for future research.

The additional constraint on the heuristic value for the initial state improves the overall coverage whenever used. Its use is orthogonal to the disambiguation technique and it turned out that such constraint in combination with disambiguations and the optimization for all syntactic states results in the best performing variant of the potential heuristic. This variant outperforms the LM-Cut, merge-and-shrink, and flow heuristics solving 90, 106, and 185 more problems, respectively, on the standard benchmark set. However, it is still surpassed by all four best-performing non-portfolio planners from the last IPC 2018.

Acknowledgements

The work of Antonín Komenda and Rostislav Horčík was supported by the Czech Science Foundation (grant no. 18-24965Y). The work of Daniel Fišer was supported by the Czech Science Foundation (grant no. 19-22555Y). The experimental evaluation was supported by the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics".

References

Alcázar, V., and Torralba, Á. 2015. A reminder about the importance of computing and exploiting invariants in planning. In *Proc. ICAPS'15*, 2–6.

Alcázar, V.; Borrajo, D.; Fernández, S.; and Fuentetaja, R. 2013. Revisiting regression in planning. In *Proc. IJCAI'13*, 2254–2260.

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS^+ planning. *Computational Intelligence* 11(4):625–655.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1–2):5–33.

Bonet, B., and van den Briel, M. 2014. Flow-based heuristics for optimal planning: Landmarks and merges. In *Proc. ICAPS'14*, 47–55.

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

Chen, Y.; Xing, Z.; and Zhang, W. 2007. Long-distance mutual exclusion for propositional planning. In *Proc. IJCAI'07*, 1840–1845.

Eriksson, S.; Röger, G.; and Helmert, M. 2018. A proof system for unsolvable planning tasks. In *Proc. ICAPS'18*, 65–73.

Fišer, D., and Komenda, A. 2018. Fact-alternating mutex groups for classical planning. *Journal of Artificial Intelligence Research* 61:475–521.

Fišer, D.; Torralba, Á.; and Shleyfman, A. 2019. Operator mutexes and symmetries for simplifying planning tasks. In *Proc. AAAI'19*, 7586–7593.

Fišer, D. 2020. Lifted fact-alternating mutex groups and pruned grounding of classical planning problems. In *Proc. AAAI'20*, to appear.

Franco, S.; Torralba, Á.; Lelis, L. H.; and Barley, M. 2017. On creating complementary pattern databases. In *Proc. IJCAI'17*.

Franco, S.; Lelis, L. H. S.; Barley, M.; Edelkamp, S.; Martinez, M.; and Moraru, I. 2018. The Complementary1 planner in IPC 2018. In *IPC 2018 planner abstracts*, 28–31.

Franco, S.; Lelis, L. H. S.; and Barley, M. 2018. The Complementary2 planner in IPC 2018. In *IPC 2018 planner abstracts*, 32–36.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *Proc. AAAI'07*, 1007–1012.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proc. ICAPS'09*, 162–169.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge & shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the Association for Computing Machinery* 61(3):16.1–16.63.

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

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

Huang, R.; Chen, Y.; and Zhang, W. 2012. SAS+ planning as satisfiability. *Journal of Artificial Intelligence Research* 43:293–328.

Moraru, I.; Edelkamp, S.; Martinez, M.; and Franco, S. 2018. Planning-pdbs planner. In *IPC 2018 planner abstracts*, 69–73.

Pommerening, F., and Helmert, M. 2015. A normal form for classical planning tasks. In *Proc. ICAPS'15*, 188–192.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015a. From non-negative to general operator cost partitioning. In *Proc. AAAI'15*, 3335–3341.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015b. From non-negative to general operator cost partitioning: Proof details. Technical Report CS-2014-005, University of Basel, Department of Mathematics and Computer Science.

Pommerening, F.; Helmert, M.; and Bonet, B. 2017. Higher-dimensional potential heuristics for optimal classical planning. In *Proc. AAAI'17*, 3636–3643.

Rintanen, J.; Heljanko, K.; and Niemelä, I. 2006. Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence* 170(12-13):1031–1080.

Seipp, J., and Helmert, M. 2018. Counterexample-guided Cartesian abstraction refinement for classical planning. *Journal of Artificial Intelligence Research* 62:535–577.

Seipp, J.; Pommerening, F.; Röger, G.; and Helmert, M. 2016. Correlation complexity of classical planning domains. In *Proc. IJCAI'16*, 3242–3250.

Seipp, J.; Pommerening, F.; and Helmert, M. 2015. New optimization functions for potential heuristics. In *Proc. ICAPS'15*, 193–201.

Seipp, J. 2018. Fast downward scorpion. In *IPC 2018 planner abstracts*, 77–79.

Sievers, S.; Wehrle, M.; and Helmert, M. 2016. An analysis of merge strategies for merge-and-shrink heuristics. In *Proc. ICAPS'16*, 294–298.

van den Briel, M.; Benton, J.; Kambhampati, S.; and Vossen, T. 2007. An LP-based heuristic for optimal planning. In *Proc. CP'07*, 651–665.