Potential Heuristics: Weakening Consistency Constraints

Pascal Lauer[1,2], **Daniel Fišer**[3]

[1]Saarland Informatics Campus, Saarland University, Saarbrücken, Germany
[2]School of Computing, The Australian National University, Canberra, Australia
[3]Aalborg University, Denmark
lauer@cs.uni-saarland.de, danfis@danfis.cz

Abstract

In classical planning, admissible potential heuristics are computed by solving linear programs (LPs) with constraints expressing consistency and goal-awareness of the heuristic. Potential heuristics can return negative estimates. So, given a potential heuristic h^{P}, the actual heuristic used in search is another heuristic defined as $h^{\mathsf{P}}_{0+}(s)=\max(h^{\mathsf{P}}(s),0)$ for every reachable state s. In this paper, we reformulate the LP constraints for consistency of h^{P} so that they ensure consistency of h^{P}_{0+} instead. This leads to more informative heuristics with positive impact on the overall performance in exchange for a more time and memory demanding computation using mixed integer linear programs instead of LPs.

1 Introduction

Potential heuristics (Pommerening et al. 2015; Pommerening, Helmert, and Bonet 2017) are a family of heuristics in classical planning that assign a numerical value, called potential, to each fact (or sets of facts). The heuristic value for a given state is simply a sum of potential of facts (or sets of facts) appearing in the state. So far, all methods for finding potentials of admissible potential heuristics are based on solving linear programs (LPs) where admissibility is ensured by constraints expressing goal-awareness in all reachable goal states and consistency over all reachable transitions. Such formulations allow to obtain different kinds of potential heuristics by changing objective functions of the LP (Seipp, Pommerening, and Helmert 2015; Fišer, Horčík, and Komenda 2020). A distinctive feature of potential heuristics is that they are allowed to return negative values, because it allows to find a more informative potential heuristics. However, when used during search, negative values are treated as zeros. More precisely, we first construct a potential heuristic h^{P} that can return negative estimates for some states. But then, during the search, we use a different heuristic h^{P}_{0+} defined as $h^{\mathsf{P}}_{0+}(s) = \max(h^{\mathsf{P}}(s), 0)$ for every reachable state s.

In this paper, we focus precisely on this aspect of potential heuristics. Since h^{P}_{0+} is the heuristic that is used during the search, it is not necessary for h^{P} to be consistent. However, all previous methods for finding potential heuristics enforce consistency of h^{P}. Here, we show how to reformulate the LP constraints ensuring consistency of h^{P} so that they ensure consistency of h^{P}_{0+} instead. This way, we can lose the consistency of h^{P}, but it does not matter in practice because h^{P}_{0+} remains consistent and goal-aware and h^{P}_{0+} is the heuristic that is used during the search. More importantly, the weakening of consistency constraints that we propose allows to find more informative potential heuristics. Unfortunately, ensuring consistency of h^{P}_{0+} is more computationally demanding than ensuring consistency of h^{P}, because it requires solving a mixed integer linear program instead of LP. However, we show experimentally that it still yields a significantly better performance in optimal planning with potential heuristics maximizing heuristic values for initial states, and a slightly better performance for other variants of potential heuristics when used in a simple portfolio planner.

2 Background

An **FDR planning task** (Bäckström and Nebel 1995) is a tuple $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$. $\mathcal{V}$ is a finite set of **variables**, each $v \in \mathcal{V}$ has a finite **domain** $\mathrm{dom}(v)$. A **fact** $\langle v, x \rangle$ is a pair of $v \in \mathcal{V}$ and $x \in \mathrm{dom}(v)$. $\mathcal{F} = \{\langle v, x \rangle \mid v \in \mathcal{V}, x \in \mathrm{dom}(v)\}$ is the set of all facts, $\mathcal{F}_v = \{\langle v, x \rangle \mid x \in \mathrm{dom}(v)\}$ is the set of facts of variable v, and similarly for sets of variables $V \subseteq \mathcal{V}$: $\mathcal{F}_V = \bigcup_{v \in V} \mathcal{F}_v$. Given $p \subseteq \mathcal{F}$, $\mathcal{V}(p)$ denotes all variables appearing in p, i.e., $\mathcal{V}(p) = \{v \mid \langle v, x \rangle \in p\}$.

A **partial state** $p \subseteq \mathcal{F}$ is a set of facts such that there is at most one fact of each variable, i.e., $|p \cap \mathcal{F}_v| \leq 1$ for every $v \in \mathcal{V}$. A partial state s is called **state** if $|s| = |\mathcal{V}|$. I is an **initial state**. G is a partial state called **goal**, and a state s is a **goal state** if $G \subseteq s$. Given partial states p and t, we say that t **extends** p if $p \subseteq t$. $\mathcal{O}$ is a finite set of **operators**, $o \in \mathcal{O}$ is defined by its partial states precondition $\mathrm{pre}(o)$ and effect $\mathrm{eff}(o)$, and a cost $\mathrm{cost}(o) \in \mathbb{R}_0^+$. We assume $\mathrm{pre}(o) \cap \mathrm{eff}(o) = \emptyset$. $o \in \mathcal{O}$ is **applicable** in a state s if $\mathrm{pre}(o) \subseteq s$. The resulting state of this application is $o[\![s]\!] = (s \setminus \mathcal{F}_{\mathcal{V}(\mathrm{eff}(o))}) \cup \mathrm{eff}(o)$. A sequence of operators $\pi = \langle o_1, \ldots, o_n \rangle$ is applicable in a state s_0 if there are states $s_1, \ldots, s_n$ s.t. 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 is $\pi[\![s_0]\!] = s_n$ and $\mathrm{cost}(\pi) = \sum_{i=1}^{n} \mathrm{cost}(o_i)$ is the cost of π. A sequence of operators π is called an **s-plan** if π is applicable in s and $\pi[\![s]\!]$ is a goal state. I-plans are simply called **plans**. An s-plan is called **optimal** if its cost is minimal among all s-plans.

A state s is called **reachable** if there exists a sequence

of operators π applicable in I s.t. $\pi[\![I]\!] = s$. $\mathcal{R}$ denotes the set of all reachable states in Π. An operator o is **reachable** if it is applicable in some reachable state. A triple $\langle s, o, s' \rangle$, denoted as $s \xrightarrow{o} s'$, is called **reachable transition** if $s \in \mathcal{R}$, $o \in \mathcal{O}$ is applicable in s and $s' = o[\![s]\!]$. A state s is a **dead-end** if $G \not\subseteq s$ and there is no s-plan. A **heuristic** is a function $h : \mathcal{R} \mapsto \mathbb{R} \cup \{\infty\}$ estimating 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. A heuristic h is called (a) **forward admissible** (f-admissible) if $h(s) \leq h^\star(s)$ for every $s \in \mathcal{R}$; (b) **forward goal-aware** (f-goal-aware) if $h(s) \leq 0$ for every reachable goal state s; and (c) **forward consistent** (f-consistent) if $h(s) \leq h(o[\![s]\!]) + \mathrm{cost}(o)$ for all $s \in \mathcal{R}$ and $o \in \mathcal{O}$ applicable in s.

We define heuristics over reachable states (instead of all states) because we intend to use them in a forward search. The definition also helps improve heuristic values (h-values) by state invariants describing (an overapproximation of) the reachable state space. We allow negative heuristic values, as is usual in literature on potential heuristics, because it allows to find more informative potential heuristics. It is well-known that (forward) goal-aware and (forward) consistent heuristics are also (forward) admissible.

A mutex is a set of facts that is not part of any reachable state, i.e., M is a **mutex** if $M \not\subseteq s$ for every $s \in \mathcal{R}$. The most obvious mutex is a fact pair of the same variable, but more can be found by computing so-called fam-groups (Helmert 2009; Fišer 2020, 2023; Fišer and Komenda 2018), or using the h^m heuristic (Bonet and Geffner 2001; Alcázar and Torralba 2015). Fišer, Horčík, and Komenda (2020) showed that using mutexes can significantly improve informativeness of potential heuristics when used to compute disambiguations:

Let $v \in \mathcal{V}$ denote a variable, and let p denote a partial state. A set of facts $X \subseteq \mathcal{F}_v$ is called a **disambiguation of** v **for** p if for every $s \in \mathcal{R}$ s.t. $p \subseteq s$ it holds that $X \cap s \neq \emptyset$.

A disambiguation of a variable v for a partial state p is a set of facts $X \subseteq \mathcal{F}_v$ from the same variable v such that every reachable state extending p contains a fact from X. So, a disambiguation of v for p allows us to filter out facts of the variable v that cannot be part of any reachable state extending p. Disambiguations can be used for finding unreachable operators and determining unsolvability of tasks. If, for some operator $o \in \mathcal{O}$, a disambiguation of some $v \in \mathcal{V}$ for $\mathrm{pre}(o)$ is empty, then o is unreachable; and if a disambiguation of some $v \in \mathcal{V}$ for G is empty, the task is unsolvable. So, from now on, we will only consider tasks with non-empty disambiguations of the goal and the operators' preconditions.

For notational convenience, we use the following **disambiguation maps** $\mathcal{D}$: Given a variable $v \in \mathcal{V}$, $\mathcal{D}_G(v)$ denotes a disambiguation of v for G. Given an operator $o \in \mathcal{O}$ and $v \in \mathcal{V}(\mathrm{eff}(o))$, $\mathcal{D}_o(v)$ denotes a disambiguation of v for $\mathrm{pre}(o)$. Fišer et al. (2020) provide a detailed description of disambiguations and how to use mutexes to compute them.

3 Potential Heuristics

Potential heuristics (Pommerening et al. 2015), sometimes also called *atomic*, map each fact to a numerical value called *potential*, and the h-value for a given state s is the sum of potentials of all facts appearing in s. Higher-dimensional potential heuristics (Pommerening, Helmert, and Bonet 2017) generalize this idea from single facts to sets of facts. Recently, Fišer and Steinmetz (2024) showed that higher-dimensional potential heuristics can be computed as atomic potential heuristics using certain compilations where sets of facts are explicitly represented as single facts. So, here we consider only atomic potential heuristics as all ideas we present here directly translate to higher-dimensional potential heuristics via the aforementioned compilations.

Definition 1. A **potential function** is a function $\mathtt{P} : \mathcal{F} \mapsto \mathbb{R}$.

A **potential heuristic** for $\mathtt{P}$, denoted as $h^{\mathtt{P}}$, is defined as $h^{\mathtt{P}}(s) = \sum_{f \in s} \mathtt{P}(f)$ for every reachable state $s \in \mathcal{R}$.

All known methods for finding potential functions are based on solving a linear program (LP) with constraints expressing goal-awareness and consistency of the resulting potential heuristics. Here, we consider the constraints introduced by Fišer, Horčík, and Komenda (2020, Theorem 7) adapted to our notation, because they often induce more informative potential heuristics while maintaining forward goal-awareness and forward consistency.

Theorem 2. *Let* $\mathtt{P}$ *denote a potential function, and let* $\mathcal{D}$ *denote disambiguation maps. If*

$$\sum_{v \in \mathcal{V}} \max_{f \in \mathcal{D}_G(v)} \mathtt{P}(f) \leq 0 \tag{1}$$

and

$$\sum_{v \in \mathcal{V}(\mathrm{eff}(o))} \max_{f \in \mathcal{D}_o(v)} \mathtt{P}(f) - \sum_{f \in \mathrm{eff}(o)} \mathtt{P}(f) \leq \mathrm{cost}(o) \tag{2}$$

for every operator $o \in \mathcal{O}$*, then* $h^{\mathtt{P}}$ *is forward goal-aware, forward consistent and forward admissible.*

It follows from Theorem 2 that a potential function $\mathtt{P}$ inducing forward admissible $h^{\mathtt{P}}$ can be obtained as any solution to the LP with variables $\mathtt{P}(f)$ for all facts $f \in \mathcal{F}$, the constraint Eq. 1 ensuring forward goal-awareness, and the constraint Eq. 2 for each operator $o \in \mathcal{O}$ ensuring forward consistency. So, the objective function of this LP can be freely chosen and there are many different ways how to do that (Pommerening et al. 2015; Seipp, Pommerening, and Helmert 2015; Fišer, Horčík, and Komenda 2020).

The resulting $h^{\mathtt{P}}$ can return negative estimates for some states which is interpreted during search as simply returning zero. In other words, the actual heuristic function (which we denote as $h^{\mathtt{P}}_{0+}$) used during search returns, for each state s, the maximum of $h^{\mathtt{P}}(s)$ and zero: Given a potential heuristic $h^{\mathtt{P}}$, $h^{\mathtt{P}}_{0+}$ denotes another heuristic function defined as $h^{\mathtt{P}}_{0+}(s) = \max(h^{\mathtt{P}}(s), 0)$ for every reachable state $s \in \mathcal{R}$.

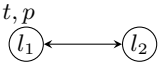
4 Weakening Consistency Constraints

In this section, we show that we can further weaken the consistency constraint Eq. 2 so that $h^{\mathtt{P}}$ (and more importantly $h^{\mathtt{P}}_{0+}$) is (possibly) more informative and remains forward admissible. The idea is quite simple: Since the actual heuristic function used in search is $h^{\mathtt{P}}_{0+}$ and not $h^{\mathtt{P}}$, it is enough to ensure forward goal-awareness and forward consistency of $h^{\mathtt{P}}_{0+}$, i.e., $h^{\mathtt{P}}$ does not need to be forward consistent.

$\mathcal{V} = \{t, p\}$, $\mathrm{dom}(t) = \{l_1, l_2\}$, $\mathrm{dom}(p) = \{l_1, l_2, t\}$

$I = \{\langle t, l_1\rangle, \langle p, l_1\rangle\}$

$G = \{\langle p, l_2\rangle\}$



$o \in \mathcal{O}$	$\mathrm{pre}(o)$	$\mathrm{eff}(o)$	$\mathrm{cost}(o)$
$\mathrm{drive}(x, y)$	$\{\langle t, x\rangle\}$	$\{\langle t, y\rangle\}$	10
$\mathrm{pickup}(z)$	$\{\langle t, z\rangle, \langle p, z\rangle\}$	$\{\langle p, t\rangle\}$	1
$\mathrm{drop}(z)$	$\{\langle t, z\rangle, \langle p, t\rangle\}$	$\{\langle p, z\rangle\}$	1

for $(x, y) \in \{(l_1, l_2), (l_2, l_1)\}$, $z \in \{l_1, l_2\}$

Figure 1: Example logistics planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G\rangle$ with one truck t, one package p and two locations l_1 and l_2.

Consider the simple logistics example task depicted in Figure 1 with one truck t, one package p and two locations l_1 and l_2. For any potential function $\mathtt{P}$ obeying the constraints from Theorem 2, the maximum possible heuristic value of $h^{\mathtt{P}}$ for the initial state is $h^{\mathtt{P}}(I)$=2 (e.g., $\mathtt{P}(\langle t, l_1\rangle)$=$\mathtt{P}(\langle t, l_2\rangle)$=0, $\mathtt{P}(\langle p, l_1\rangle)$=2, $\mathtt{P}(\langle p, l_2\rangle)$=0, $\mathtt{P}(\langle p, t\rangle)$=1): Since the goal-awareness constraint (Eq. 1) is $\max_{x \in \mathrm{dom}(t)} \mathtt{P}(\langle t, x\rangle) + \mathtt{P}(\langle p, l_2\rangle) \leq 0$, it is possible to have $h^{\mathtt{P}}(I) > 2$ only if $|\mathtt{P}(\langle p, l_1\rangle) - \mathtt{P}(\langle p, l_2\rangle)| > 2$. But the consistency constraints (Eq. 2) over pickup and drop actions imply $|\mathtt{P}(\langle p, l_1\rangle) - \mathtt{P}(\langle p, l_2\rangle)| \leq 2$.

However, there exist potential functions $\mathtt{P}$ such that $h^{\mathtt{P}}$ is forward admissible and $h^{\mathtt{P}}(I)$=$h^\star(I)$=12. For example, $\mathtt{P}(\langle t, l_1\rangle)$=0, $\mathtt{P}(\langle t, l_2\rangle)$=$-10$, $\mathtt{P}(\langle p, l_1\rangle)$=12, $\mathtt{P}(\langle p, l_2\rangle)$ = -12, $\mathtt{P}(\langle p, t\rangle)$=11. This $\mathtt{P}$ violates the consistency constraint Eq. 2 for the operator $\mathrm{drop}(l_2)$. Therefore $h^{\mathtt{P}}$ is not forward consistent, but it turns out $h^{\mathtt{P}}_{0+}$ is forward consistent. In the following, we show how to find such potential functions maintaining forward consistency of $h^{\mathtt{P}}_{0+}$ by weakening the consistency constraints (Eq. 2) from Theorem 2.

We start with a lemma showing a simple reformulation of consistency criteria for heuristics of the form $\max(h(s), 0)$ that will be the basis for our new consistency constraints.

Lemma 3. *Let h denote a heuristic function, and let $s \xrightarrow{o} s'$ denote a reachable transition. Then*

$$\max(h(s), 0) - \max(h(s'), 0) \leq \mathrm{cost}(o)$$

if and only if

$$\min(h(s) - h(s'), h(s)) \leq \mathrm{cost}(o).$$

Proof. Since $\mathrm{cost}(o)$ is non-negative, both inequalities hold for any negative value of $h(s)$. For $h(s) \geq 0$, we have that $\max(h(s), 0) - \max(h(s'), 0) = h(s) - \max(h(s'), 0) = h(s) + \min(-h(s'), 0) = \min(h(s) - h(s'), h(s))$. □

For notational convenience, we define the following two shorthands. Given disambiguation maps $\mathcal{D}$ and an operator $o \in \mathcal{O}$, we use $C^{\mathrm{con}}_{\mathcal{D}}(o)$ to denote the left-hand side of Eq. 2:

$$C^{\mathrm{con}}_{\mathcal{D}}(o) = \sum_{V \in \mathcal{V}(\mathrm{eff}(o))} \max_{f \in \mathcal{D}_o(V)} \mathtt{P}(f) - \sum_{f \in \mathrm{eff}(o)} \mathtt{P}(f), \qquad (3)$$

and we use $C^{\mathrm{pre}}_{\mathcal{D}}(o)$ to denote an upper bound on the h-values of $h^{\mathtt{P}}$ in reachable states where o is applicable:

$$C^{\mathrm{pre}}_{\mathcal{D}}(o) = \sum_{v \in \mathcal{V}} \max_{f \in \mathcal{D}_o(v)} \mathtt{P}(f). \qquad (4)$$

Note that in case of $C^{\mathrm{pre}}_{\mathcal{D}}(o)$, we use disambiguation to consider only (an overapproximation of) possible *reachable* states extending $\mathrm{pre}(o)$. So since we sum over all variables, the h-values of $h^{\mathtt{P}}$ in *reachable* states where o is applicable can be at most the value of $C^{\mathrm{pre}}_{\mathcal{D}}(o)$.

Finally, we can formulate the main contribution of this paper in the following theorem.

Theorem 4. *Let $\mathtt{P}$ denote a potential function, and let $\mathcal{D}$ denote disambiguation maps. If Eq. 1 holds and*

$$\min(C^{\mathrm{con}}_{\mathcal{D}}(o), C^{\mathrm{pre}}_{\mathcal{D}}(o)) \leq \mathrm{cost}(o) \qquad (5)$$

for every operator $o \in \mathcal{O}$, then $h^{\mathtt{P}}$ is f-admissible and $h^{\mathtt{P}}_{0+}$ is f-consistent, f-goal-aware and f-admissible.

Proof. Eq. 1 ensures f-goal-awareness of $h^{\mathtt{P}}$ and therefore also of $h^{\mathtt{P}}_{0+}$. Let $s \xrightarrow{o} s'$ denote a reachable transition. We have that $h^{\mathtt{P}}(s) - h^{\mathtt{P}}(s') \leq C^{\mathrm{con}}_{\mathcal{D}}(o)$, and $h^{\mathtt{P}}(s) \leq C^{\mathrm{pre}}_{\mathcal{D}}(o)$. Therefore we have that $\min(h^{\mathtt{P}}(s) - h^{\mathtt{P}}(s'), h^{\mathtt{P}}(s)) \leq \min(C^{\mathrm{con}}_{\mathcal{D}}(o), C^{\mathrm{pre}}_{\mathcal{D}}(o)) \leq \mathrm{cost}(o)$. Therefore it follows from Lemma 3 that $h^{\mathtt{P}}_{0+}(s) - h^{\mathtt{P}}_{0+}(s') \leq \mathrm{cost}(o)$, and therefore $h^{\mathtt{P}}_{0+}$ is f-consistent, and therefore also f-admissible. Lastly, since for every reachable state $s \in \mathcal{R}$ we have that $h^{\mathtt{P}}(s) \leq h^{\mathtt{P}}_{0+}(s)$, it follows that also $h^{\mathtt{P}}$ is f-admissible. □

Theorem 4 shows that we can replace Eq. 2 with Eq. 5 and the resulting $h^{\mathtt{P}}$ will remain forward admissible while $h^{\mathtt{P}}_{0+}$ will also be forward consistent (i.e., A* with $h^{\mathtt{P}}_{0+}$ does not need to re-open states). Moreover, since $C^{\mathrm{con}}_{\mathcal{D}}(o)$ is the left-hand side of Eq. 2, the constraint Eq. 5 can only be weaker than Eq. 2. This means that for a fixed objective function, the maximum objective value for constraints from Theorem 4 is at least as high as for constraints from Theorem 2.

Note that we can also mix the consistency constraints from Theorems 2 and 4, i.e., we can use Eq. 2 for some operators and Eq. 5 for others. For instance in our example task, we can obtain $\mathtt{P}$ such that $h^{\mathtt{P}}(I) = h^\star(I) = 12$ by using Eq. 5 only for $\mathrm{drop}(l_2)$ and Eq. 2 for all other operators. However, it is not clear how to decide between Eq. 2 and 5 for each operator so that the resulting potential heuristic is improved. Here, we compare only variants where all consistency constraints are either Eq. 2 or Eq. 5.

The remaining question is how to implement the constraint Eq. 5 in practice. One option is to use a mixed-integer linear program (MIP) with a big-M formulation: A constraint $\min(a, b) \leq c$ can be reformulated as a pair of constraints $a - xM \leq c$ and $b - (1-x)M \leq c$ where x is a binary *indicator* variable and M is a large constant. In our experiments, we found that it is very hard to find a suitable value of M in a *domain-independent* manner. However, we used the CPLEX solver which provides so called indicator constraints which allow to easily formulate the constraint Eq. 5. Nevertheless, it requires having one additional binary variable per operator and therefore it is much harder to solve these MIPs than the original LPs with constraints from Theorem 2. So, we run the solver with a time limit and use the possibly suboptimal solutions found within that time limit.

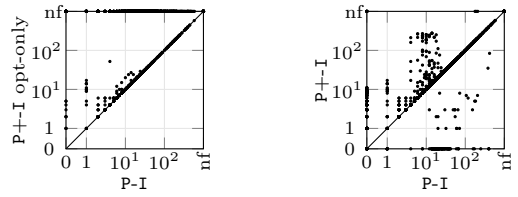


Figure 2: Per-task comparison of h-values for initial states; "nf": tasks where no potentials were found, "opt-only": only values proved optimal within 30 minutes time limit.

5 Experimental Evaluation

The method was implemented[1] in C and evaluated on a cluster with Intel Xeon E5-2650v3 processors. We use all planning domains from the optimal track of International Planning Competitions (IPCs) from 1998 to 2023 excluding the ones containing conditional effects after translation. We merge, for each domain, all benchmarks from different IPCs resulting in 54 domains and 1 806 tasks overall. 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 mutex groups inference proposed by Fišer (2020, 2023).

We use 30 minutes time and 8 GB memory limit for each task. We run A* (Hart, Nilsson, and Raphael 1968) with the baseline potential heuristics, denoted as P, computed using constraints from Theorem 2, and our method, P+, using Theorem 4. We also use a simple portfolio, por, where we first run P and if P terminates after t minutes without finding a plan, then we run P+ with only $30-t$ minutes time limit (and 8 GB memory limit). We consider three variants of potential heuristics: optimized for the maximum h-value of the initial state (I) (Pommerening et al. 2015), maximum average h-value for all syntactic states (A) (Seipp, Pommerening, and Helmert 2015), and the variant A with enforced maximum h-value for the initial state (AI) (Fišer, Horčík, and Komenda 2020). Since P+ often fails to find an optimal solution of the MIP within 30 minutes, we set the time limit for the MIP solver to 2 minutes and use the suboptimal solution if available (otherwise, we abstain from solving the task).

Figure 2 shows per-task comparison of h-values for initial states when maximizing for that value. P+ found higher h-values for initial states in 314 tasks spread over 21 domains, and lower h-values in 142 tasks in 20 domains (data points below diagonal in Figure 2 (right)). The lower h-values are due to using suboptimal solutions of MIPs as the optimal MIP solutions for P+ are guaranteed to dominate the solutions for P. The MIP solver was able to provide optimal solutions within 30 minutes only for 512 tasks, and all of them were obtained also within 2 minutes. Moreover, using the MIP solver with 2 minutes time limit allowed to obtain potential functions in 1 236 more tasks.

Table 1 shows the number of solved tasks (coverage) per domain and overall. We can observe a significantly higher coverage for the I variant. Interestingly, in most tasks solved by P+-I but not by P-I, the h-value for the initial state is the same for both P and P+. So the question is how much can different LP/MIP solutions with the same objective value influence the performance of the planner and whether there is a way to select the best one, which should be investigated in future. (Note that P+ only widens the range of possibilities.)

domain	I			A			AI		
	P	P+	por	P	P+	por	P	P+	por
blocks (35)	21	**28**	**28**	28	28	28	28	28	28
caldera (20)	10	**12**	10	12	12	12	12	12	12
driverlog (20)	9	**13**	**13**	**13**	11	**13**	13	13	13
elevators (50)	31	31	31	31	**35**	**35**	31	**33**	**33**
freecell (80)	48	**60**	**60**	37	37	37	**74**	69	**74**
ged (20)	16	**18**	**18**	15	15	15	15	**18**	**18**
logistics (63)	13	**23**	**23**	24	24	24	24	24	24
mprime (35)	18	**20**	19	25	25	25	25	25	25
nomystery (20)	10	**14**	**14**	**14**	13	**14**	14	14	14
parking (40)	1	**7**	**7**	**16**	6	**16**	**16**	7	**16**
pipesw-notank (50)	25	**26**	**26**	25	25	25	**30**	28	**30**
recharging-robots (20)	**13**	12	**13**	**13**	12	**13**	**13**	11	**13**
scanalyzer (50)	25	25	25	23	23	23	**27**	25	**27**
snake (20)	**15**	14	**15**	**14**	13	**14**	**16**	13	**16**
spider (20)	**14**	11	**14**	**14**	11	**14**	**16**	12	**16**
tidybot (40)	**32**	30	**32**	**32**	30	**32**	**32**	30	**32**
tpp (30)	6	**8**	**8**	6	6	6	8	8	8
trucks (30)	9	**14**	**14**	**14**	11	**14**	14	14	14
visitall (40)	25	**30**	**30**	25	**28**	**28**	27	**30**	**30**
woodworking (50)	19	**27**	**27**	**27**	25	**27**	31	31	31
zenotravel (20)	8	**11**	**11**	11	11	11	11	11	11
others (1053)	558	**566**	565	569	566	**570**	**584**	582	**584**
Σ (1806)	926	1 000	**1 003**	988	967	**996**	1 061	1 038	**1 069**

Table 1: Number of solved tasks; "others": sum over domains where difference between P/P+/por was at most 1 for all variants I/A/AI.

In case of A and AI, using P+ is mostly detrimental. The reason seems to be that the MIP solver struggles to optimize for the objective function averaging over all syntactic states (higher time limits did not lead to better results). In case of A, out of 30 tasks solved by P but not by P+, solutions of P+ had lower objective value in 18 tasks. In case of AI, out of 33 tasks solved by P but not by P+, P+ ended up with lower objective value in 19 tasks and it could not find any solution in 6 tasks. However, the simple portfolio por tends to get the best of both P and P+, increasing the overall coverage for all variants of potential heuristics.

6 Conclusion and Future Work

Up until now, all methods for computing potential functions P were based on enforcing consistency of the resulting potential heuristics $h^{\mathtt{P}}$. Here, we introduce an alternative constraint ensuring consistency of heuristics maximizing over $h^{\mathtt{P}}$ and zero, which are the heuristics actually used in the search instead of $h^{\mathtt{P}}$. This opens up the space of available potential heuristics in practice and it often leads to more informative heuristics. Nevertheless, these new constraints require solving MIP instead of LP, which is much more computationally demanding in practice. This could be mitigated by using the new constraints only for some operators. Finding how to select these operators is left for future research. Another question open for future is how to apply our method to the so-called operator potential heuristics (Fišer, Torralba, and Hoffmann 2024) in the context of symbolic search.

[1] https://gitlab.com/danfis/cpddl, branch icaps25-pot-hmax0

Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project ID 232722074 – SFB 1102.

References

Alcázar, V.; and Torralba, Á. 2015. A Reminder about the Importance of Computing and Exploiting Invariants in Planning. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, 2–6.

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.

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 Hoffmann, J. 2024. Boosting Optimal Symbolic Planning: Operator-Potential Heuristics. *Artificial Intelligence*, 104174.

Fišer, D. 2020. Lifted Fact-Alternating Mutex Groups and Pruned Grounding of Classical Planning Problems. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI'20)*, 9835–9842.

Fišer, D. 2023. Operator Pruning Using Lifted Mutex Groups via Compilation on Lifted Level. In *Proceedings of the 33rd International Conference on Automated Planning and Scheduling (ICAPS'23)*, 118–127.

Fišer, D.; Horčík, R.; and Komenda, A. 2020. Strengthening Potential Heuristics with Mutexes and Disambiguations. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS'20)*, 124–133.

Fišer, D.; and Steinmetz, M. 2024. Towards Feasible Higher-Dimensional Potential Heuristics. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS'24)*, 210–220.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Helmert, M. 2009. Concise Finite-Domain Representations for PDDL Planning Tasks. *Artificial Intelligence*, 173: 503–535.

Pommerening, F.; Helmert, M.; and Bonet, B. 2017. Higher-Dimensional Potential Heuristics for Optimal Classical Planning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*, 3636–3643.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015. From Non-Negative to General Operator Cost Partitioning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, 3335–3341.

Seipp, J.; Pommerening, F.; and Helmert, M. 2015. New Optimization Functions for Potential Heuristics. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, 193–201.