Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling (ICAPS 2022)

Operator-Potentials in Symbolic Search: From Forward to Bi-directional Search

Daniel Fišer[1,2], **Álvaro Torralba**[3], **Jörg Hoffmann**[1]

[1] Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
[2] Czech Technical University in Prague, Faculty of Electrical Engineering, Czech Republic
[3] Aalborg University, Denmark
danfis@danfis.cz, alto@cs.aau.dk, hoffmann@cs.uni-saarland.de

Abstract

Symbolic search using binary decision diagrams is a state-of-the-art technique for cost-optimal planning. Heuristic search in this context has been problematic as even a very informative heuristic can be detrimental in case it induces difficult-to-represent state partitionings. It was recently shown that operator-potential heuristics can address this issue in forward search by computing a numeric potential for each operator corresponding to the change of the heuristic value induced by that operator.

Forward search is, however, not the best known variant of symbolic search. Here we investigate the integration with backward and bi-directional search instead. We prove that forward search (distance-to-goal) operator-potential heuristics can be turned into backward search (distance-to-initial-state) heuristics elegantly in this context, by summing the backward search path operator-potentials with the initial state goal-distance estimate. We run exhaustive experiments on IPC benchmarks, showing that significant performance improvements can be obtained over symbolic forward search and other state-of-the-art techniques.

1 Introduction

In cost-optimal planning, A* search with admissible heuristics and symbolic search are two complementary planning techniques. Admissible heuristics aim at reducing the number of explored states during search by estimating the cost-to-go from each state (e.g., Haslum and Geffner 2000; Edelkamp 2001; Helmert and Domshlak 2009; Helmert et al. 2014). Symbolic search utilizes binary decision diagrams (BDDs) (Bryant 1986) for an efficient representation of sets of states which greatly improves an exhaustive (blind) search as it allows to represent a large fraction of a state space efficiently (Edelkamp and Reffel 1998; Edelkamp 2002). Combination of admissible heuristics with symbolic search is, unfortunately, not straightforward. Although there has been some success with symbolic pattern databases (e.g., Kissmann and Edelkamp 2011; Franco et al. 2017; Torralba, López, and Borrajo 2018), it was also shown that even very informative heuristics can be detrimental in the symbolic search (Speck, Geißer, and Mattmüller 2020) as the heuristics need to be not only informative, but they also need to induce a good partitioning of the state space so that sets of states are efficiently represented as BDDs.

Recently, Fišer, Torralba, and Hoffmann (2022) showed how to integrate potential heuristics (Pommerening et al. 2015) into symbolic forward search. Potential heuristics assign a numeric value (a potential) to each fact of the planning task, and the sum of the potentials of the facts true in a state is an admissible heuristic estimate. Fišer, Torralba, and Hoffmann (2022) showed that potentials of facts can be easily transformed into operator-potentials, i.e., instead of assigning a numerical value to each fact, we can associate each operator with a numerical value (operator-potential) corresponding to the change of heuristic value induced by the operator. It turns out, operator-potential heuristics can be easily integrated into the symbolic search using a method called GHSETA* (Jensen, Veloso, and Bryant 2008). In GHSETA*, operators are partitioned into transition relations (TRs) by both their costs and the change of heuristic values they induce. This results in a good partitioning of sets of states during search which in turn significantly improves performance in the symbolic *forward* search.

Forward search is, however, not the best known variant of symbolic search. Here, we focus on the backward and bi-directional search instead. This is non-trivial because, to use a forward-search heuristic function h in backward search, h needs to be "reversed". How this can be done differs widely depending on the kind of heuristic function. A generally applicable technique is to reverse the planning task instead, by enumerating individual goal states or by the transformation to transition normal form (TNF) (Pommerening and Helmert 2015). Yet the former is obviously ineffective, and Fišer, Torralba, and Hoffmann (2022) report that using TNF is often detrimental for symbolic search.

Our key insight here is that the operator-potentials are constructed in a way so that, if we split a plan π into two sub-sequences π' and π'', then the goal-distance estimate plus the sum of operator-potentials over π'' is a lower bound on the cost of π'. Hence symbolic backward search with operator-potentials can be done analogously to forward search. This in turn extends to symbolic *bi-directional* search where we can choose any combination of operator-potential or blind heuristics for each search direction.

We run exhaustive experiments on IPC benchmarks, evaluating a large range of algorithm variants, considering mul-

tiple search directions, and heuristic functions. Regarding operator-potential heuristics in backward search, the results show that these can improve performance in many domains, though overall those are counter-balanced by losses in other domains. However, the combination of bi-directional search with potential heuristics turns out to be a clear win. The best-performing configuration in our experiments combines forward symbolic search using an operator-potential heuristic with blind backward search. This manages to leverage the benefits of bi-directional search where that is strong, while suffering from only small losses where it is not. Consequently this configuration beats the previous state of the art in overall coverage.

2 Preliminaries

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 $\mathrm{cost}(o) \in \mathbb{R}_0^+$. 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 $V \in \mathcal{V} \setminus \mathrm{vars}(\mathrm{eff}(o))$. We also assume that for every $V \in \mathrm{vars}(\mathrm{pre}(o)) \cap \mathrm{vars}(\mathrm{eff}(o))$ it holds that $\mathrm{pre}(o)[V] \neq \mathrm{eff}(o)[V]$.

Given non-negative integers $k, n \in \mathbb{N}_0$, $[k, n]$ denotes the set $\{k, \ldots, n\}$ for $k \leq n$, and $[k, n]$ is defined as an empty set for $k > n$. Moreover, $[n]$ denotes a shorthand for $[1, n]$. 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 $i \in [n]$. The resulting state of this application is $\pi[\![s_0]\!] = s_n$ and $\mathrm{cost}(\pi) = \sum_{i=1}^{n} \mathrm{cost}(o_i)$ 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 set of facts $M \subseteq \mathcal{F}$ is a **mutex** if $M \not\subseteq s$ for every reachable state $s \in \mathcal{R}$.

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 $h(s) \leq h(o[\![s]\!]) + \mathrm{cost}(o)$ 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. In the context of heuristic search, h-value of a state node s refers to the heuristic value of s, g-value to the cost of the sequence of operators leading to s, and f-value is a sum of g-value and the maximum of h-value and zero (since we allow negative h-values).

3 Background on Potential and Operator-Potential Heuristics

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

Definition 1. 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., $h^{\mathtt{P}}(s) = \sum_{f \in s} \mathtt{P}(f)$.

We leverage prior work on disambiguation (Alcázar et al. 2013) to strengthen potential heuristics (Fišer, Horčík, and Komenda 2020). A disambiguation of a variable V for a set of facts p is a set of facts $F \subseteq \mathcal{F}_V$ from V such that every reachable state extending p contains one of the facts from F.

Definition 2. 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., $\langle V, s[V] \rangle \in F$).

Moreover, we say that a mapping $\mathcal{D} : (\mathcal{O} \times \mathcal{V}) \cup \mathcal{V} \mapsto 2^{\mathcal{F}}$ is a **disambiguation map** if (i) for every operator $o \in \mathcal{O}$ and every variable $V \in \mathrm{vars}(\mathrm{eff}(o))$ it holds that $\mathcal{D}(o, V) \subseteq \mathcal{F}_V$ is a disambiguation of V for $\mathrm{pre}(o)$ such that $|\mathcal{D}(o, V)| \geq 1$; and (ii) for every variable $V \in \mathcal{V}$ it holds that $\mathcal{D}(V) \subseteq \mathcal{F}_V$ is a disambiguation of V for G such that $|\mathcal{D}(V)| \geq 1$.

Clearly, if the disambiguation of V for p is an empty set (for any V), then all states extending p are unreachable. Therefore, we can use empty disambiguations to determine unsolvability of planning tasks (if G extends p), or to prune unreachable operators (if a precondition of the operator extends p). So, from now on we will consider only non-empty disambiguations in a form of disambiguation maps, which simplifies the notation.

Now we can state sufficient conditions for the potential heuristic to be admissible, which we will need later on.

Theorem 3. *(Fišer, Horčík, and Komenda 2020) 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 let $\mathcal{D}$ denote a disambiguation map. If*

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

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

$$\sum_{V \in \text{vars}(\text{eff}(o))} \max_{f \in \mathcal{D}(o,V)} \mathtt{P}(f) - \sum_{f \in \text{eff}(o)} \mathtt{P}(f) \leq \text{cost}(o), \quad (2)$$

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

Potentials for the admissible potential heuristic can be obtained by solving a linear program (LP) consisting of constraints corresponding to conditions from Theorem 3, i.e., $\mathtt{P}(f)$ for each $f \in \mathcal{F}$ corresponds to an LP variable and any solution satisfying Eq. (1) and Eq. (2) gives us potentials for an admissible potential heuristic.

Recently, Fišer, Torralba, and Hoffmann (2022) introduced operator-potential heuristics that associate potentials not with facts but with operators. An operator-potential $\mathtt{Q}(o)$ for an operator o is defined as a left-hand side of Eq. (2) with the opposite sign. The heuristic value for a state s reached by a sequence of operators π is then the heuristic value for the initial state of the corresponding potential heuristic ($\sum_{f \in I} \mathtt{P}(f)$) plus the sum of operator-potentials for operators in π.

Moreover, it was shown that if the potential heuristic for $\mathtt{P}$ is admissible and, for each operator $o \in \mathcal{O}$, preconditions on all variables affected by o are known exactly, then the corresponding operator-potential heuristic for $\mathtt{Q}$ is also admissible and the heuristic values are equal for all reachable states.

Definition 4. Given a potential function $\mathtt{P}$, and a disambiguation map $\mathcal{D}$, a function $\mathtt{Q} : \mathcal{O} \mapsto \mathbb{R}$ is called an **operator-potential function** for $\mathtt{P}$ and $\mathcal{D}$ if

$$\mathtt{Q}(o) = \sum_{f \in \text{eff}(o)} \mathtt{P}(f) - \sum_{V \in \text{vars}(\text{eff}(o))} \max_{f \in \mathcal{D}(o,V)} \mathtt{P}(f) \quad (3)$$

for every operator $o \in \mathcal{O}$.

Given an operator-potential function $\mathtt{Q}$ for $\mathtt{P}$ and $\mathcal{D}$ such that $|\mathcal{D}(o,V)| = 1$ for every $o \in \mathcal{O}$ and every $V \in \text{vars}(\text{eff}(o))$, an **operator-potential heuristic** $h^{\mathtt{Q}} : \mathcal{R} \mapsto \mathbb{R} \cup \{\infty\}$ for $\mathtt{Q}$ is defined as $h^{\mathtt{Q}}(s) = \sum_{f \in I} \mathtt{P}(f) + \sum_{i \in [n]} \mathtt{Q}(o_i)$ for any sequence of operators $\pi = \langle o_1, \ldots, o_n \rangle$ such that $\pi[\![I]\!] = s$.

Theorem 5. *(Fišer, Torralba, and Hoffmann 2022) Let $\mathcal{D}$ denote a disambiguation map such that $|\mathcal{D}(o,V)| = 1$ for every $o \in \mathcal{O}$ and every $V \in \text{vars}(\text{eff}(o))$, let $\mathtt{P}$ denote a potential function, and let $\mathtt{Q}$ denote an operator-potential function for $\mathtt{P}$ and $\mathcal{D}$. Then $h^{\mathtt{Q}}$ is well-defined, and $h^{\mathtt{Q}}(s) = h^{\mathtt{P}}(s)$ for every reachable state s, and $h^{\mathtt{Q}}$ is admissible (goal-aware, consistent) if $h^{\mathtt{P}}$ is admissible (goal-aware, consistent).*

Note that the operator-potential heuristics are guaranteed to be consistent only if the corresponding potential heuristics are consistent, and $|\mathcal{D}(o,v)| = 1$ for every $o \in \mathcal{O}$ and every $V \in \text{vars}(\text{eff}(o))$, i.e., the preconditions on variables affected by each operator are known exactly. This may require a transformation of the planning task. The experimental evaluation by Fišer, Torralba, and Hoffmann (2022) showed that the best-performing transformation is simply enumerating all possible combinations of facts from all disambiguations $\mathcal{D}(o,V)$ such that $|\mathcal{D}(o,V)| > 1$ for all operators' preconditions.

As we discuss in the next section, for a smooth integration of operator-potentials in symbolic search, the operator-potentials $\mathtt{Q}(o)$ need to be integer values. Fišer, Torralba, and Hoffmann (2022) showed that the naïve rounding of operator-potentials to the nearest smaller integer values can result in a path-dependent (and thus inconsistent) heuristics which can have a detrimental effect on the symbolic search. However, this problem can be resolved by constructing a mixed-integer linear program (MIP) whose solution are integer-valued operator potentials. So, instead of using LP with constraints Eq. (1) and Eq. (2), we can construct a MIP with constraints Eq. (1), Eq. (2), and Eq. (3), where $\mathtt{P}(f)$ for each fact $f \in \mathcal{F}$ corresponds to a real-valued variable, and $\mathtt{Q}(o)$ for each operator $o \in \mathcal{O}$ correspond to an integer-valued variable. Therefore, the solution to such MIP will give us integer-valued operator potentials.

4 Background on Symbolic Search with Operator-Potentials

Explicit state-space search operates on individual states, whereas symbolic search (McMillan 1993) works on sets of states represented by their characteristic functions. A characteristic function f_S of a set of states S is a Boolean function assigning 1 to states that belong to S and 0 otherwise. Binary Decision Diagrams (BDDs) (Bryant 1986) are an efficient data-structure to represent Boolean functions as directed acyclic graphs. The size of a BDD is the number of nodes in this representation. The main advantage of using BDDs is that often a BDD is much smaller than the number of states it represents. In fact, BDDs can be exponentially smaller (Edelkamp and Kissmann 2008), and most operations take only polynomial time in the size of the BDD.

The most prominent implementation of symbolic heuristic search in the context of automated planning is BDDA* (Edelkamp and Reffel 1998) which is a variant of A* (Hart, Nilsson, and Raphael 1968) using BDDs to represent sets of states. In BDDA*, operators of planning tasks are represented as transition relations, also using BDDs. A *transition relation* (TR) of an operator o is a characteristic function of pairs of states $(s, o[\![s]\!])$ for all states s such that o is applicable in s. Having a TR T_o for every operator $o \in \mathcal{O}$, we can construct a TR of a set of operators with the same cost c as $T_c = \bigvee_{o \in \mathcal{O}, \text{cost}(o)=c} T_o$. As the size of T_c may be exponential in the number of operators with cost c, in practice, it is often a good idea to use *disjunctive partitioning* to keep the size at bay (Jensen, Veloso, and Bryant 2008; Torralba et al. 2017). Moreover, mutexes can be used for a more accurate approximation of reachable states (Torralba et al. 2017).

Like A*, BDDA* expands states in ascending order of their f-value. To take advantage of the symbolic representation, BDDA* represents all states with the same g and h value in a single BDD $S_{g,h}$ (disjunctive partitioning of $S_{g,h}$ can also be used). Given a set of states $S_{g,h}$ and a TR T_c, $\text{image}(S_{g,h}, T_c)$ computes the set of successor states reachable from any state in $S_{g,h}$ by applying any operator rep-

resented by T_c.[1] The g-value of the resulting set of successor states is simply $g + c$. These successor states have to be split according to their h value. This can usually be performed efficiently (e.g., with symbolic PDBs (Kissmann and Edelkamp 2011)) by representing the heuristic as a BDD S_h per heuristic value that represents the set of states with that value and performing a conjunction.

GHSETA* (Jensen, Veloso, and Bryant 2008) encodes the heuristic function as a part of TRs by partitioning operators not only by their cost but also by their impact on heuristic values. If such partitioning is possible, i.e., it is possible to precompute how each operator changes heuristic values, then GHSETA* provides a very efficient way of evaluating heuristics within symbolic search. As it turns out, this is the case for operator-potential heuristics (Fišer, Torralba, and Hoffmann 2022).

In particular, instead of creating a TR T_c for all operators o having $\mathrm{cost}(o) = c$, we can create a TR $T_{c,q}$ representing all operators o such that $\mathrm{cost}(o) = c$ and $\mathtt{Q}(o) = q$. For the initial state, the g-value is set to zero, and the h-value is set to $\sum_{f \in I} \mathtt{P}(f)$. For all subsequent states $S_{g,h}$ expanded by the TR $T_{c,q}$, the g-value and h-value of the resulting state $S'_{g',h'} = \mathrm{image}(S_{g,h}, T_{c,q})$ is set to $g' = g + c$ and $h' = h + q$, respectively.

Note that floating-point $\mathtt{Q}(o)$ values may result in many small TRs that differ only slightly in their $\mathtt{Q}(o)$ values, which in turn can lead to different BDDs for every state in the search, greatly reducing the efficacy of the symbolic search. This is why it is desirable to have integer-valued $\mathtt{Q}(o)$ values as discussed in the previous section. Moreover, Fišer, Torralba, and Hoffmann (2022) showed that GHSETA* can also be adapted to work with the path-dependent (and thus possibly inconsistent) variant of the operator-potential heuristic. In that case, we need to allow re-opening states that were previously closed with a higher g-value by maintaining multiple BDDs representing closed states, each corresponding to a specific g-value.

The symbolic backward search differs from the forward search only in that we switch the initial and goal states and use the function $\mathrm{pre\text{-}image}$ instead of image for computing successor sets of states. That is: (a) the backward search starts from the BDD representing goal states and proceeds towards the initial state; (b) the search terminates once a BDD containing the initial state is popped from the priority queue; and (c) we use the function $\mathrm{pre\text{-}image}(S, T)$ that for the given set of states S and TR T returns the set of states S' such that $\mathrm{image}(S', T) = S$.

The symbolic bi-directional search combines the forward and backward search described above. It maintains a separate priority queue for each direction, and at each step it decides whether to expand a set of states in forward or backward direction. The decision is made based on the estimate how long it takes to make the next step and the one with the smaller estimate is taken. The estimate is computed based on the size of the BDD and time needed for its expansion in the previous step.

[1]We do not go into details here and in the rest of this section, but Torralba et al. (2017) provide a detailed description.

The search terminates with the extraction of a plan when a set of states S_f with the g-value g_f is found in the forward direction, and a set of states S_b with the g-value g_b is found in the backward direction, and $S_f \cap S_b \neq \emptyset$, and $g_f + g_b < m_f + m_b$ where m_f and m_g denote the minimal g-value of all sets of states in the open list in forward and backward direction, respectively.

Note that if different operator-potentials are used in the forward and backward direction, then they require a separate partitioning of operators into TRs, i.e., in that case different sets of TRs must be used for each search direction.

5 Operator-Potential Heuristics for Backward Search

So far, we summarized prior research showing that it is possible to transform any potential function into an operator-potential function and compute the heuristic value for the state s reached in a forward search by a sequence of operators π by summing the operator-potentials of operators in π. In this section, we focus on operator-potential heuristics for the backward search, i.e., the question is whether we can use a similar approach to compute admissible heuristic values for states that are reached by a search starting in goal state(s) going backwards towards the initial state.

One straightforward way to do it might be to construct a dual (reverse) planning task (Massey 1999; Pettersson 2005; Suda 2013) and compute the potentials there. The resulting heuristic would be a lower bound on the distance from the initial state (of the reversed task) to each state, which can be used to guide the backward search. In terms of the LP used for the computation of potential functions (assuming $|\mathcal{D}(o, V)| = 1$ for every $o \in \mathcal{O}$ and every $V \in \mathrm{vars}(\mathrm{eff}(o))$), this corresponds to switching the operands on the left hand side of Eq. (2) and replacing Eq. (1) with the constraint $\sum_{f \in I} \mathtt{P}(f) \leq 0$. This seems like a simple transformation, because the resulting potentials would give us an admissible heuristic for the backward search for any objective function.

However, the problem is that often the goal G does not specify a single goal state, so it is not clear how to efficiently compute the heuristic value for a set of goal states as a starting point of the backward search. One option could be to partition the set of goal states according to their heuristic value, i.e., by the sum of their potentials. Unfortunately, this is unfeasible in general, as there may be exponentially many goal states, and potentially each of those could have a different heuristic value. Another option might be to use a compilation to the transition normal form (TNF) (Pommerening and Helmert 2015), where the goal always describes a single state. But, as reported by Fišer, Torralba, and Hoffmann (2022), using TNF is often detrimental for the symbolic search.

Here, we show that, surprisingly, operator-potentials provide an even simpler solution to this problem. It turns out that the operator-potentials computed for the forward search can also be used for the backward search without any change. More precisely, assuming the potential function $\mathtt{P}$ satisfies the conditions from Theorem 3 (and therefore $h^{\mathtt{P}}$ is consistent, goal-aware, and admissible) and $|\mathcal{D}(o, V)| = 1$

for every $o \in \mathcal{O}$ and every $V \in \text{vars}(\text{eff}(o))$, we show that:

(i) The heuristic value for the initial state $h^{\mathtt{Q}}(I) = \sum_{f \in I} \mathtt{P}(f)$ is an admissible heuristic value for any goal state $s_g \supseteq G$ in backward search. This is simply because $h^{\mathtt{Q}}(I)$ is a lower bound on the cost of the cheapest path from the initial state to the closest goal state.

(ii) In the following Theorem 6 and Corollary 7, we show that for any pair of sequences of operators $\pi_1 = \langle o_1, \ldots, o_k \rangle$ and $\pi_2 = \langle o_{k+1}, \ldots, o_n \rangle$ such that $\pi_1[\![I]\!] = s$ and $\pi_2[\![s]\!] \supseteq G$ is a goal state, it holds that $h^{\mathtt{Q}}(I) + \sum_{i \in [k+1,n]} \mathtt{Q}(o_i) \leq \sum_{i \in [k]} \text{cost}(o_i)$. Or in other words, $h^{\mathtt{Q}}(I) + \sum_{i \in [k+1,n]} \mathtt{Q}(o_i)$ is an admissible heuristic value for any state s reached in the backward search by a sequence of operators $\langle o'_n, \ldots, o'_{k+1} \rangle$, where o'_i denotes the reverse counterpart of o_i.

The main result of this section formulated in Theorem 6 is based on the following two observations. First, $\mathtt{Q}(o)$ is equal to the left hand side of Eq. (2) with opposite sign, and Eq. (2) is expressing a consistency of the corresponding potential function. Therefore, it follows that $-\mathtt{Q}(o) \leq \text{cost}(o)$ for every operator o. Second, the goal-awareness ensures that for any plan π it holds that $h^{\mathtt{Q}}(I) + \sum_{o \in \pi} \mathtt{Q}(o) \leq 0$. So, if we split π into two sub-sequences π' and π'', then $h^{\mathtt{Q}}(I) + \sum_{o \in \pi'} \mathtt{Q}(o) \leq -\sum_{o \in \pi''} \mathtt{Q}(o) \leq \sum_{o \in \pi''} \text{cost}(o)$. Therefore, $h^{\mathtt{Q}}(I) + \sum_{o \in \pi'} \mathtt{Q}(o)$ is a lower bound on $\sum_{o \in \pi''} \text{cost}(o)$, i.e., we can use operator-potentials over the sub-sequence π' for the computation of the lower bound on the cost of the sub-sequence π''.

Theorem 6. *Let $\mathcal{D}$ denote a disambiguation map such that $|\mathcal{D}(o, V)| = 1$ for every $o \in \mathcal{O}$ and every $V \in \text{vars}(\text{eff}(o))$, let $\mathtt{P}$ denote a potential function such that Eq. (1) and Eq. (2) hold, let $\mathtt{Q}$ denote an operator-potential function for $\mathtt{P}$ and $\mathcal{D}$ and $h^{\mathtt{Q}}$ the corresponding operator-potential heuristic, and let $\pi = \langle o_1, \ldots, o_n \rangle$ denote a plan. Then $h^{\mathtt{Q}}(I) + \sum_{i \in N} \mathtt{Q}(o_i) \leq \sum_{i \in [n] \setminus N} \text{cost}(o_i)$ for any $N \subseteq [n]$.*

Proof. Since Eq. (1) and Eq. (2) hold for $\mathtt{P}$, it follows from Theorem 3 that $h^{\mathtt{P}}$ is consistent and goal-aware. And since $|\mathcal{D}(o, V)| = 1$ for every $o \in \mathcal{O}$ and every $V \in \text{vars}(\text{eff}(o))$, it follows from Theorem 5 that $h^{\mathtt{Q}}$ is also consistent and goal-aware. So, for $N = \emptyset$, the theorem holds, because $h^{\mathtt{Q}}(I) = \sum_{f \in I} \mathtt{P}(f) \leq \sum_{i \in [n]} \text{cost}(o_i)$ is an admissible estimate for the initial state.

For $1 \leq |N| < n$: From the goal-awareness of $h^{\mathtt{Q}}$, it follows that $h^{\mathtt{Q}}(I) + \sum_{i \in [n]} \mathtt{Q}(o_i) = h^{\mathtt{Q}}(I) + \sum_{i \in N} \mathtt{Q}(o_i) + \sum_{i \in [n] \setminus N} \mathtt{Q}(o_i) \leq 0$, and therefore

$$h^{\mathtt{Q}}(I) + \sum_{i \in N} \mathtt{Q}(o_i) \leq -\sum_{i \in [n] \setminus N} \mathtt{Q}(o_i).$$

Furthermore, from Definition 4 (Eq. (3)) and Eq. (2) it follows that

$$\mathtt{Q}(o) = \sum_{f \in \text{eff}(o)} \mathtt{P}(f) - \sum_{V \in \text{vars}(\text{eff}(o))} \max_{f \in \mathcal{D}(o,V)} \mathtt{P}(f) \geq \text{cost}(o)$$

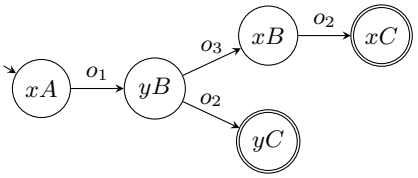
and therefore $-\mathtt{Q}(o) \leq \text{cost}(o)$ for every operator o. So it follows that $-\sum_{i \in [n] \setminus N} \mathtt{Q}(o_i) \leq \sum_{i \in [n] \setminus N} \text{cost}(o_i)$, and

$\mathcal{V} = \{v_1, v_2\}$, $\text{dom}(v_1) = \{x, y\}$, $\text{dom}(v_2) = \{A, B, C\}$

$I = \{\langle v_1, x \rangle, \langle v_2, A \rangle\}$, $G = \{\langle v_2, C \rangle\}$

$o \in \mathcal{O}$	pre(o)	eff(o)	cost(o)	$\mathtt{Q}(o)$
o_1	$\langle v_1, x \rangle, \langle v_2, A \rangle$	$\langle v_1, y \rangle, \langle v_2, B \rangle$	1	-1
o_2	$\langle v_2, B \rangle$	$\langle v_2, C \rangle$	1	-1
o_3	$\langle v_1, y \rangle$	$\langle v_1, x \rangle$	1	-1

f	$\mathtt{P}(f)$
$\langle v_1, x \rangle$	0
$\langle v_1, y \rangle$	1
$\langle v_2, A \rangle$	2
$\langle v_2, B \rangle$	0
$\langle v_2, C \rangle$	-1



$h^{\mathtt{P}}(I) = h^{\mathtt{Q}}(I) = 2$

Figure 1: Example planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$ showing that operator-potential heuristics can be inconsistent for backward search even if $|\mathcal{D}(o, V)| = 1$ for every $o \in \mathcal{O}$ and every $V \in \text{vars}(\text{eff}(o))$. If we run the backward search and we use the same heuristic value $h^{\mathtt{Q}}(I) = 2$ for both goal states "xC" and "yC", then we get a different heuristic value for the state "yB"—the path from "xC" results in the value $h^{\mathtt{Q}}(I) + \mathtt{Q}(o_2) + \mathtt{Q}(o_3) = 0$, but the path from "$yC$" results in the value $h^{\mathtt{Q}}(I) + \mathtt{Q}(o_2) = 1$.

therefore

$$h^{\mathtt{Q}}(I) + \sum_{i \in N} \mathtt{Q}(o_i) \leq -\sum_{i \in [n] \setminus N} \mathtt{Q}(o_i) \leq \sum_{i \in [n] \setminus N} \text{cost}(o_i),$$

which concludes the proof. □

Corollary 7. *Let $\mathtt{Q}$, $h^{\mathtt{Q}}$, and π be as before. Then $h^{\mathtt{Q}}(I) + \sum_{i \in [k+1,n]} \mathtt{Q}(o_i) \leq \sum_{i \in [k]} \text{cost}(o_i)$ for every $k \in [n]$.*

Proof. Follows directly from Theorem 6 as it is a special case of $N = [k+1, n]$ (and thus $[n] \setminus N = [k]$). □

Let s be a state reached by a sequence of operators $\langle o'_1, \ldots, o'_n \rangle$ in the *backward* search. It follows from Corollary 7 that if s is reachable also in the forward direction, then setting the heuristic value for s to $h^{\mathtt{Q}}(I) + \sum_{i \in [n]} \mathtt{Q}(o'_i)$ gives us an admissible estimate in the backward search. Moreover, the same heuristic value can be used even if s is not reachable in the forward direction, because in that case s is a dead-end in the backward direction, and therefore any heuristic value is admissible.

Since we intend to use the operator-potential heuristics in the backward *symbolic* search, the remaining question to be answered is whether these heuristics are path-dependent (inconsistent) in this context, i.e., whether we need to re-open states during the backward search from a set of all goal states. Unfortunately, the heuristics can be path-dependent if the goal G specifies more than one goal state. Intuitively, the set of goal states $\mathcal{G} = \{s \supseteq G \mid s \in \mathcal{R}\}$ can contain goal states with very different minimal distances to the initial state, but we assign all of them the same heuristic value $h^{\mathtt{Q}}(I)$. So, if there exists a state s (backwards) reachable from two different goal states $s_g, s'_g \in \mathcal{G}$, then we can

Domain	$\overleftarrow{\texttt{b}}$	$\overleftarrow{\texttt{I}}$	$\overleftarrow{\texttt{A+I}}$	$\overleftarrow{\texttt{S}_{1k}\texttt{+I}}$	$\overleftarrow{\texttt{M}_2\texttt{+I}}$	$\overrightarrow{\overleftarrow{\texttt{b}}\texttt{-b}}$	$\overrightarrow{\texttt{A+I}}\overleftarrow{\texttt{b}}$	$\overrightarrow{\texttt{M}_2\texttt{+I}}\overleftarrow{\texttt{b}}$	$\overrightarrow{\texttt{S}_{1k}\texttt{+I}}\overleftarrow{\texttt{b}}$	$\overrightarrow{\texttt{A+I}}\overleftarrow{\texttt{A+I}}$	$\overrightarrow{\texttt{A+I}}\overleftarrow{\texttt{M}_2\texttt{+I}}$	$\overrightarrow{\texttt{b}}$	$\overrightarrow{\texttt{A+I}}$	scrp	comp2	$\texttt{pot}_{\texttt{A+I}}$	ms	lmc
agricola (20)	6	3	4	2	4	**20**	17	19	14	13	15	17	16	6	10	3	4	0
airport (50)	23	25	25	23	23	26	27	25	27	27	25	23	27	**39**	28	35	21	27
barman (34)	13	6	6	6	6	**16**	13	13	14	10	12	**16**	14	11	15	11	11	8
blocks (35)	24	32	23	22	23	**33**	31	31	31	32	32	22	31	28	31	28	21	28
caldera (20)	16	16	16	15	14	17	17	16	17	17	17	**18**	17	13	15	12	12	12
childsnack (20)	0	0	0	0	0	**5**	**5**	**5**	**5**	**5**	**5**	4	**5**	0	2	0	0	0
data-network (20)	8	3	4	4	4	13	13	13	12	13	13	11	13	**14**	13	9	12	13
depot (22)	4	5	4	4	4	8	10	10	10	11	11	6	11	**14**	8	11	10	7
driverlog (20)	9	10	10	10	10	14	14	14	13	14	14	11	14	**15**	**15**	13	13	14
elevators (50)	16	10	10	10	10	43	43	43	43	43	42	35	35	**44**	**44**	31	31	40
floortile (40)	**34**	27	31	31	31	**34**	**34**	**34**	**34**	31	31	18	17	16	33	11	16	**34**
freecell (80)	20	29	21	21	21	27	68	67	68	68	68	20	68	**72**	31	**72**	20	15
ged (20)	11	10	10	10	10	**20**	**20**	**20**	19	**20**	**20**	15	15	**20**	**20**	15	15	19
hiking (20)	15	11	12	14	12	18	15	15	16	15	15	16	15	15	**20**	14	14	11
logistics (63)	19	24	19	19	19	25	28	28	27	28	28	21	28	**37**	28	24	25	26
mprime (35)	12	11	13	10	13	16	16	17	17	16	16	27	30	**31**	24	24	23	25
mystery (30)	9	8	9	8	9	10	11	11	11	11	11	15	**19**	**19**	16	18	17	17
nomystery (20)	13	17	14	13	14	18	**20**	**20**	16	**20**	**20**	12	19	**20**	**20**	14	14	16
openstacks (100)	80	76	76	76	76	86	**91**	**91**	**91**	**91**	**91**	86	**91**	55	74	57	51	51
parcprinter (50)	41	35	33	33	32	37	46	43	44	37	37	40	48	**50**	43	47	41	39
parking (40)	0	4	0	0	0	6	15	15	15	14	14	0	13	**16**	5	**16**	2	9
pegsol (50)	22	27	25	26	25	48	48	48	48	48	48	46	48	**50**	48	48	46	48
petri-net-align (20)	16	1	1	1	1	**19**	15	15	13	8	8	15	11	0	**19**	13	7	9
pipesw-notank (50)	10	10	10	10	10	17	24	22	25	24	24	17	25	26	25	**30**	23	18
pipesw-tank (50)	6	6	6	7	6	15	19	18	19	**21**	**21**	17	20	18	19	19	16	13
rovers (40)	13	11	12	12	12	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	10	13	8	7	9
satellite (36)	10	7	10	10	10	**11**	**11**	**11**	**11**	**11**	**11**	7	10	10	10	6	6	7
scanalyzer (50)	21	23	23	23	23	21	23	23	23	23	23	21	23	**33**	22	23	23	31
snake (20)	0	0	0	0	0	7	11	11	11	11	11	7	11	**15**	14	**15**	9	7
sokoban (50)	43	38	38	38	38	48	49	49	49	**50**	49	48	**50**	**50**	48	**50**	**50**	**50**
spider (20)	0	0	0	0	0	7	13	11	11	13	13	7	13	**16**	13	**16**	6	11
storage (30)	14	14	14	14	14	15	**16**	**16**	**16**	**16**	**16**	15	**16**	**16**	15	**16**	15	15
termes (20)	9	7	7	8	7	**18**	**18**	**18**	17	16	16	12	12	14	16	12	12	6
tetris (17)	6	10	11	10	11	12	16	16	16	16	16	9	16	13	13	**17**	11	9
tidybot (40)	8	5	8	5	6	21	25	25	23	25	25	28	34	35	**39**	32	31	30
tpp (30)	8	8	8	8	8	8	12	12	12	12	12	8	12	8	**14**	8	7	7
transport (70)	17	13	13	13	13	33	33	33	31	30	29	27	24	**38**	37	24	24	23
trucks (30)	10	12	12	10	12	13	**18**	**18**	16	**18**	**18**	13	16	17	15	14	10	13
visitall (40)	16	18	19	18	19	18	22	22	23	22	22	18	22	30	**33**	30	29	18
woodwork (50)	49	33	36	36	36	49	47	48	49	46	46	38	46	**50**	48	29	29	39
zenotravel (20)	8	8	8	8	8	11	**13**	**13**	12	**13**	**13**	9	**13**	**13**	**13**	11	11	**13**
others (175)	126	126	126	126	126	**128**	127	127	127	127	127	127	127	115	127	114	114	114
Σ (1697)	785	739	727	714	720	1 025	**1 128**	1 120	1 110	1 100	1 099	936	1 109	1 112	1 096	1 000	859	901

Table 1: Number of solved tasks for selected domains and planning techniques; "others" sums results over domains with a small difference between symbolic search methods.

get two different heuristic values depending on whether we reach s from s_g or s'_g, because the path from s_g to s may differ from the path from s'_g to s. A full example is provided in Figure 1.

The application of operator-potential heuristics in symbolic backward search is straightforward. We can compute operator-potential heuristics as described in Section 3 and use it in the GHSETA* algorithm as described in Section 4. However, we must treat the heuristic as inconsistent, i.e., we need to apply re-opening of states. And similarly for symbolic bi-directional search, where we can combine any operator-potential heuristics for forward and backward direction, but, as already mentioned in Section 4, we need to use a different partitioning of operators into transition relations if the heuristics for the backward and forward direction differ.

6 Experimental Evaluation

We implemented GHSETA* with operator-potential heuristics in C.[2] 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). We used all planning domains from the optimal track of International Planning Competitions (IPCs) from 1998 to 2018 excluding the ones containing conditional effects after translation. We merged, for each domain, all benchmark suites across different IPCs. This leaves 48 domains overall.

We used a cluster of computing nodes with Intel Xeon Scalable Gold 6146 processors and CPLEX (I)LP solver v12.10. The time and memory limits were set to 30 minutes and 8 GB, respectively. We used a time limit of 30 seconds

[2]https://gitlab.com/danfis/cpddl, branch icaps22-symba-op-pot

	$\overleftarrow{\texttt{b}}$	$\overleftarrow{\texttt{I}}$	$\overleftarrow{\texttt{A+I}}$	$\overleftarrow{\texttt{M}_2\texttt{+I}}$	$\overleftarrow{\texttt{S}_{1k}\texttt{+I}}$	tot
$\overleftarrow{\texttt{b}}$	–	**19**	**16**	**18**	**20**	785
$\overleftarrow{\texttt{I}}$	13	–	8	10	**13**	739
$\overleftarrow{\texttt{A+I}}$	10	**12**	–	**4**	**11**	727
$\overleftarrow{\texttt{M}_2\texttt{+I}}$	9	**12**	0	–	**9**	720
$\overleftarrow{\texttt{S}_{1k}\texttt{+I}}$	7	8	4	6	–	714

Table 2: Summary of domain coverage for symbolic backward search. A value in row x and column y is the number of domains where x solved more tasks than y, it is bold if higher than the value in row y and column x. "tot" shows overall number of solved tasks.

for applying mutexes on the goal BDD and 10 seconds for merging transition relation BDDs (Torralba et al. 2017).

The symbolic search with operator-potentials was evaluated on tasks transformed by the method described in Section 3 so that $|\mathcal{D}(o, v)| = 1$ for every $o \in \mathcal{O}$ and every $V \in \text{vars}(\text{eff}(o))$. We evaluated GHSETA* with the following variants of the operator-potential heuristics:

- I: maximize the $h^{\texttt{Q}}$-value of the initial state (Pommerening et al. 2015).
- A+I: maximize the $h^{\texttt{Q}}$-value for the average (syntactic) state while enforcing the maximum $h^{\texttt{Q}}(I)$ (Seipp, Pommerening, and Helmert 2015; Fišer, Horčík, and Komenda 2020).
- $\texttt{S}_{1k}$+I: maximize the $h^{\texttt{Q}}$-value for 1 000 states sampled using random walks, while enforcing the maximum $h^{\texttt{Q}}(I)$ (Seipp, Pommerening, and Helmert 2015; Fišer, Horčík, and Komenda 2020).
- $\texttt{M}_2$+I: maximize the $h^{\texttt{Q}}$-value for reachable states approximated with mutex pairs while enforcing the maximum $h^{\texttt{Q}}(I)$ (Fišer, Horčík, and Komenda 2020).

We compare these to the blind search (b). We denote forward search with $\overrightarrow{\cdot}$, and backward search by $\overleftarrow{\cdot}$. For example, the blind forward search is denoted by $\overrightarrow{\texttt{b}}$, the backward search with the heuristic A+I is denoted by $\overleftarrow{\texttt{A+I}}$, and the bi-directional search with A+I used in the forward direction, and I in the backward direction is denoted by $\overrightarrow{\texttt{A+I}}$-$\overleftarrow{\texttt{I}}$.

In the case of bi-directional search, we turn off backward search if the application of mutexes on the goal BDD takes longer than 30 seconds, i.e., in tasks where it is not possible to apply mutexes on the goal BDD within the time limit, the bi-directional search is switched to the forward search (this happened in 141 out of 1 697 tasks).

Furthermore, we compare to other state-of-the-art planners. We ran A* with the LM-Cut (lmc) heuristic (Helmert and Domshlak 2009), with the merge-and-shrink (ms) heuristic with SCC-DFP merge strategy and non-greedy bisimulation shrink strategy (Helmert et al. 2014; Sievers, Wehrle, and Helmert 2016), and with the potential heuristic ($\texttt{pot}_{\texttt{A+I}}$), i.e., a variant of A+I for A*. We further compare to two of the best-performing non-portfolio planners from IPC 2018: Complementary2 (comp2) (Franco et al. 2017; Franco, Lelis, and Barley 2018), and Scorpion (scrp) (Seipp 2018; Seipp and Helmert 2018). Table 1 shows the number of solved tasks (coverage) for all variants of the symbolic backward search, the five best-performing variants of the symbolic bi-directional search with operator-potential heuristics, and all baseline planners.

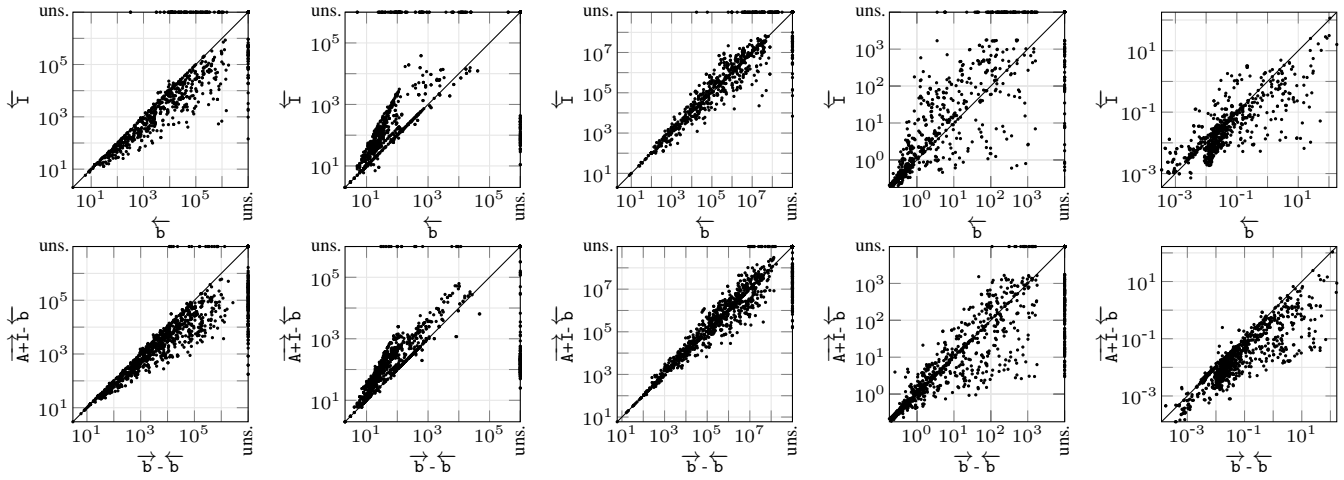
	$\emptyset$	$\overleftarrow{\texttt{b}}$	$\overleftarrow{\texttt{A+I}}$	$\overleftarrow{\texttt{I}}$	$\overleftarrow{\texttt{S}_{1k}\texttt{+I}}$	$\overleftarrow{\texttt{M}_2\texttt{+I}}$	$\overleftarrow{\text{oracle}}$
$\emptyset$	—	785	727	739	714	720	843
$\overrightarrow{\texttt{b}}$	936	1 025	950	961	934	951	1 047
$\overrightarrow{\texttt{A+I}}$	1 109	**1 128**	1 100	1 087	1 093	1 099	1 139
$\overrightarrow{\texttt{I}}$	996	1 032	985	1 000	987	988	1 058
$\overrightarrow{\texttt{S}_{1k}\texttt{+I}}$	1 081	1 110	1 073	1 066	1 062	1 074	1 121
$\overrightarrow{\texttt{M}_2\texttt{+I}}$	1 103	1 120	1 092	1 076	1 085	1 097	1 132
$\overrightarrow{\text{oracle}}$	1 131	1 151	1 111	1 103	1 112	1 114	1 162

Table 3: Overall coverage of different combinations of forward and backward search. A value in row x and column y is the overall number of solved tasks where x was used for the forward direction and y for the backward direction of the symbolic bi-directional search. $\emptyset$ means that no forward or backward search was used. oracle refers to selecting the best variant for each task for the respective search direction. The highest coverage of all non-oracle variants is in bold.

Symbolic Backward Search

The overall number of solved tasks by the symbolic backward search is not improved by utilizing operator-potential heuristics (Table 1). This is in contrast to the improvement obtained by using operator-potential heuristics in symbolic forward search as reported by Fišer, Torralba, and Hoffmann (2022) (also columns $\overrightarrow{\texttt{b}}$ and $\overrightarrow{\texttt{A+I}}$ in Table 1). However, there is still a substantial number of domains where operator-potentials are beneficial as can be observed in Table 2 comparing the number of domains where one method solved more tasks than the other. In particular, the best-performing variant $\overleftarrow{\texttt{I}}$ solved significantly more tasks in domains blocks, freecell, logistics, nomystery, pegsol, and tetris, but also significantly lacks behind the blind search ($\overleftarrow{\texttt{b}}$) in domains agricola, barman, data-network, elevators, floortile, hiking, parcprinter, petri-net-alignment, transport, and woodworking.

Similarly to the symbolic forward search, using the heuristics in backward search results in smaller BDDs representing sets of states on average (see Figure 2a(top)), and it expands more BDDs during the search (Figure 2b(top)) as expected. The reason why this behavior does not result in more solved tasks seems to be a slower expansion of BDDs in contrast to the forward search, i.e., operator-potentials do not always induce a good partitioning of symbolic states, and the overhead caused by the fact that the operator-potential heuristics in this case are path-dependent which requires reopening of states (see Figure 2d and 2e(top)). As mentioned in Section 4, re-opening states requires maintaining multiple BDDs to represent closed states, one for each g-value observed during the search. An interesting observation is that $\overleftarrow{\texttt{I}}$ seems to perform worse, compared to $\overleftarrow{\texttt{b}}$, on do-



(a) Mean number of BDD nodes per expanded BDD. (b) Number of expanded BDDs. (c) Sum of expanded BDD nodes. (d) Runtime in seconds. (e) Average runtime per expanded BDD.

Figure 2: Top row: Comparison of the symbolic backward blind search ($\overleftarrow{\texttt{b}}$) against the backward GHSETA* $\overleftarrow{\texttt{I}}$. Bottom row: Comparison of the symbolic bi-directional blind search ($\overrightarrow{\texttt{b}}$-$\overleftarrow{\texttt{b}}$) against the bi-directional GHSETA* $\overrightarrow{\texttt{A+I}}$-$\overleftarrow{\texttt{b}}$.

mains with more diverse action costs, possibly because this increases the overhead for re-opening states.

In conclusion, it seems the results with and without operator-potential heuristics are complementary, and it would be beneficial to know in advance for each task (or at least domain) whether it is better to use the operator-potential heuristics or not (Cenamor, de la Rosa, and Fernández 2016; Fawcett et al. 2014)—the overall number of solved tasks for the oracle choosing the best results from $\overleftarrow{\texttt{b}}$ and $\overleftarrow{\texttt{I}}$ for each task results in 836 solved tasks overall (843 for the oracle choosing from all variants of the symbolic backward search). We leave this question for future work.

Symbolic Bi-Directional Search

The picture significantly changes when we look at the symbolic bi-directional search. Table 3 shows the comparison of overall coverage for all evaluated combinations of forward and backward symbolic search. The best performing variant in terms of the overall coverage is $\overrightarrow{\texttt{A+I}}$-$\overleftarrow{\texttt{b}}$, i.e., the best-performing variant of the forward search (Fišer, Torralba, and Hoffmann 2022) and the blind backward search. This is not surprising given the blind backward search performs better than all variants of operator-potential heuristics in overall coverage. A similar trend can be observed also for any fixed configuration of the forward search, i.e., the blind backward search seems to be the best choice for any selection of the forward search. This can be observed also in Figure 3a comparing runtime of $\overrightarrow{\texttt{A+I}}$-$\overleftarrow{\texttt{b}}$ and $\overrightarrow{\texttt{A+I}}$-$\overleftarrow{\texttt{A+I}}$. Using blind backward search seems to be advantageous over using operator-potential heuristics, although there are few tasks where $\overleftarrow{\texttt{A+I}}$ is beneficial. The picture changes only slightly if we consider an oracle for the backward direction (1 139). Interestingly, choosing the best variant of the forward search has a bigger impact (1 151 solved tasks for blind backward search). Choosing the best variant for both forward and backward direction results in 1 162 solved tasks suggesting that better understanding which configuration fits better which task (or using portfolios) would further improve the performance of this planning technique.

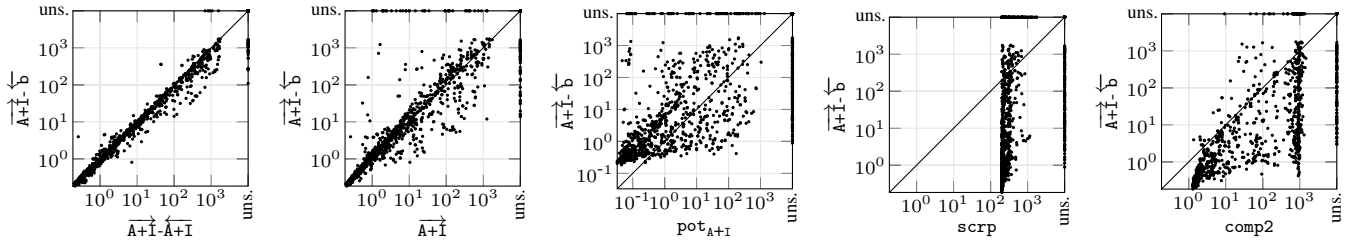
So, it seems the improved performance obtained by using operator-potentials in forward search (Fišer, Torralba, and Hoffmann 2022) translates also into the bi-directional setting. Indeed, the plot in Figure 2a(bottom) shows that the average size of BDDs is almost always smaller with operator-potentials suggesting partitioning of operators using operator-potentials is reflected in an efficient partitioning of sets of states resulting in compact representations of sets of states. The number of expanded BDDs is almost always higher for $\overrightarrow{\texttt{A+I}}$-$\overleftarrow{\texttt{b}}$ (Figure 2b(bottom)) as expected with more fine-grained partitioning of TRs using operator-potentials. This reflects on the number of BDD nodes from all expanded sets of states (Figure 2c(bottom)), which is only slightly better for the bi-directional search using operator-potentials. However, as can be seen in Figure 2e(bottom), the average time spent in expanding one BDD is almost always decreased, often resulting in smaller runtime of the whole search (Figure 2d(bottom)).

Comparison to State-of-the-Art

Table 4 summarizes the comparison to the state-of-the-art planners in terms of the overall coverage, the number of domains where one method solves more tasks than the other, and the number of tasks solved by one method but not the other. Our best variant ($\overrightarrow{\texttt{A+I}}$-$\overleftarrow{\texttt{b}}$) clearly performs better than `lmc` and `ms` in overall coverage and there are only few domains where these methods solve more tasks than $\overrightarrow{\texttt{A+I}}$-$\overleftarrow{\texttt{b}}$. $\overrightarrow{\texttt{A+I}}$-$\overleftarrow{\texttt{b}}$ also solves 128 more tasks than $\texttt{pot}_{\texttt{A+I}}$, but there

	domain dominance												task dominance												
	$\overrightarrow{\texttt{A+I}}\texttt{-}\overleftarrow{\texttt{b}}$	scrp	$\overrightarrow{\texttt{A+I}}$	$\overrightarrow{\texttt{A+I}}\texttt{-}\overleftarrow{\texttt{A+I}}$	comp2	$\overrightarrow{\texttt{b}}\texttt{-}\overleftarrow{\texttt{b}}$	$\texttt{pot}_{\texttt{A+I}}$	$\overrightarrow{\texttt{b}}$	lmc	ms	$\overleftarrow{\texttt{b}}$	$\overleftarrow{\texttt{I}}$	$\overrightarrow{\texttt{A+I}}\texttt{-}\overleftarrow{\texttt{b}}$	scrp	$\overrightarrow{\texttt{A+I}}$	$\overrightarrow{\texttt{A+I}}\texttt{-}\overleftarrow{\texttt{A+I}}$	comp2	$\overrightarrow{\texttt{b}}\texttt{-}\overleftarrow{\texttt{b}}$	$\texttt{pot}_{\texttt{A+I}}$	$\overrightarrow{\texttt{b}}$	lmc	ms	$\overleftarrow{\texttt{b}}$	$\overleftarrow{\texttt{I}}$	total
$\overrightarrow{\texttt{A+I}}\texttt{-}\overleftarrow{\texttt{b}}$	–	16	**12**	**9**	**17**	**22**	**25**	**32**	**32**	**36**	**37**	**40**	–	**131**	**59**	**33**	**104**	**123**	**200**	**224**	**266**	**309**	**348**	**390**	1 128
scrp	**21**	–	**23**	**21**	**19**	**27**	**26**	**31**	**36**	**37**	**31**	**34**	115	–	**108**	**126**	**128**	**215**	**138**	**266**	**239**	**260**	**411**	**429**	1 112
$\overrightarrow{\texttt{A+I}}$	9	13	–	8	**17**	**23**	**26**	**30**	**33**	**37**	**37**	**39**	40	105	–	**55**	**107**	**155**	**152**	**189**	**245**	**263**	**352**	**381**	1 109
$\overrightarrow{\texttt{A+I}}\texttt{-}\overleftarrow{\texttt{A+I}}$	4	15	**11**	–	16	**21**	**24**	**31**	**30**	**35**	**36**	**40**	5	114	46	–	**95**	**120**	**180**	**209**	**246**	**284**	**338**	**361**	1 100
comp2	15	16	16	**17**	–	**22**	**27**	**32**	**33**	**38**	**36**	**38**	72	112	94	91	–	**112**	**188**	**190**	**218**	**245**	**322**	**367**	1 096
$\overrightarrow{\texttt{b}}\texttt{-}\overleftarrow{\texttt{b}}$	7	14	13	10	13	–	**23**	**24**	**24**	**30**	**38**	**39**	20	128	71	45	41	–	**188**	**118**	**182**	**228**	**244**	**294**	1 025
$\texttt{pot}_{\texttt{A+I}}$	14	5	9	14	13	18	–	**23**	**24**	**22**	**29**	**28**	72	26	43	80	92	163	–	**178**	**169**	**151**	**321**	**334**	1 000
$\overrightarrow{\texttt{b}}$	6	10	7	9	7	6	17	–	17	**22**	**30**	**30**	32	90	16	45	30	29	114	–	**130**	**129**	**188**	**245**	936
lmc	5	2	4	7	6	12	14	**20**	–	**21**	**27**	**28**	39	28	37	47	23	58	70	95	–	**100**	**216**	**237**	901
ms	5	1	1	6	4	11	4	16	15	–	**24**	**26**	40	7	13	43	8	62	10	52	58	–	**191**	**218**	859
$\overleftarrow{\texttt{b}}$	2	7	3	5	4	1	11	7	15	14	–	**19**	5	84	28	23	11	4	106	37	100	117	–	**97**	785
$\overleftarrow{\texttt{I}}$	1	7	2	0	4	2	9	9	12	14	13	–	1	56	11	0	10	8	73	48	75	98	51	–	739

Table 4: Left: Number of domains where one method solved more tasks than the other. A value in (x, y) (row x, column y) is the number of domains where x solved more tasks than y, it is bold if higher than (y, x). Middle: Number of tasks solved by one method but not the other. A value in (x, y) is the number of tasks solved by x, but not by y; it is bold if higher than (y, x). Right: "total" shows overall number of solved tasks. Variants of GHSETA* introduced in this paper are highlighted.



(a) $\overrightarrow{\texttt{A+I}}\texttt{-}\overleftarrow{\texttt{b}}$ vs. $\overrightarrow{\texttt{A+I}}\texttt{-}\overleftarrow{\texttt{A+I}}$ (b) $\overrightarrow{\texttt{A+I}}\texttt{-}\overleftarrow{\texttt{b}}$ vs. $\overrightarrow{\texttt{A+I}}$ (c) $\overrightarrow{\texttt{A+I}}\texttt{-}\overleftarrow{\texttt{b}}$ vs. $\texttt{pot}_{\texttt{A+I}}$ (d) $\overrightarrow{\texttt{A+I}}\texttt{-}\overleftarrow{\texttt{b}}$ vs. scrp (e) $\overrightarrow{\texttt{A+I}}\texttt{-}\overleftarrow{\texttt{b}}$ vs. comp2

Figure 3: Per-task comparison of the runtime (in seconds) of selected planners against the best-performing variant of GHSETA* with operator-potential heuristic ($\overrightarrow{\texttt{A+I}}\texttt{-}\overleftarrow{\texttt{b}}$).

are still 14 domains where explicit state space search with the potential heuristic A+I outperforms the bi-directional symbolic search. One of the reasons seems to be the fact that $\texttt{pot}_{\texttt{A+I}}$ can be extremely fast (Figure 3c): Out of 1 000 solved tasks, $\texttt{pot}_{\texttt{A+I}}$ solves 295 of them under 100 milliseconds whereas $\overrightarrow{\texttt{A+I}}\texttt{-}\overleftarrow{\texttt{b}}$ does not solve any task within this time limit, and $\texttt{pot}_{\texttt{A+I}}$ solves 556 tasks under one second whereas $\overrightarrow{\texttt{A+I}}\texttt{-}\overleftarrow{\texttt{b}}$ solves only 430.

scrp and comp2 also seem to be complementary to our method even though we solve 16 and 28 more tasks overall, respectively. Unfortunately, the runtime comparison (Figure 3d and 3e) can be very misleading in this case, because both scrp and comp2 use a long pre-computation phase for construction of their heuristic functions.

The symbolic forward search with operator-potentials ($\overrightarrow{\texttt{A+I}}$) also seems to be complementary when comparing the number of domains where it dominates our method. The runtime comparison in Figure 3b also shows that combining $\overrightarrow{\texttt{A+I}}$ with the backward symbolic search instead of using the forward-only search is detrimental in a sizeable amount of tasks. This suggests that using a more clever way to turn off the search in the backward direction would improve the results (see also the oracle comparison in Table 3).

7 Conclusion

Combining heuristics with symbolic search is a notoriously difficult task. Recently, it was shown that potential heuristics transformed into operator-potential heuristics can be efficiently integrated into the symbolic forward search (Fišer, Torralba, and Hoffmann 2022). Here, we show that the same operator-potential heuristics can be used in the symbolic backward search which can be beneficial in a substantial number of domains. Moreover, we show that the observed synergy between operator-potential heuristics and symbolic forward search also translates into the symbolic backward bi-directional search resulting in a state-of-the-art performance over the standard benchmark set.

Acknowledgements

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—project number 389792660—TRR 248 (see https://perspicuous-computing.science). 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.; Borrajo, D.; Fernández, S.; and Fuentetaja, R. 2013. Revisiting Regression in Planning. In *Proc. IJCAI'13*, 2254–2260.

Alcázar, V.; and Torralba, Á. 2015. A Reminder about the Importance of Computing and Exploiting Invariants in Planning. In *Proc. ICAPS'15*, 2–6.

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

Bryant, R. E. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, 35(8): 677–691.

Cenamor, I.; de la Rosa, T.; and Fernández, F. 2016. The IBaCoP Planning System: Instance-Based Configured Portfolios. *Journal of Artificial Intelligence Research*, 56: 657–691.

Edelkamp, S. 2001. Planning with Pattern Databases. In *Proc. ECP'01*, 13–24.

Edelkamp, S. 2002. Symbolic Pattern Databases in Heuristic Search Planning. In *Proc. AIPS'02*, 274–283.

Edelkamp, S.; and Kissmann, P. 2008. Limits and Possibilities of BDDs in State Space Search. In *Proc. AAAI'08*, 1452–1453.

Edelkamp, S.; and Reffel, F. 1998. OBDDs in Heuristic Search. In *Proc. KI'98*, 81–92.

Fawcett, C.; Vallati, M.; Hutter, F.; Hoffmann, J.; Hoos, H.; and Leyton-Brown, K. 2014. Improved Features for Runtime Prediction of Domain-Independent Planners. In *Proc. ICAPS'14*, 355–359.

Fišer, D. 2020. Lifted Fact-Alternating Mutex Groups and Pruned Grounding of Classical Planning Problems. In *Proc. AAAI'20*, 9835–9842.

Fišer, D.; Horčík, R.; and Komenda, A. 2020. Strengthening Potential Heuristics with Mutexes and Disambiguations. In *Proc. ICAPS'20*, 124–133.

Fišer, D.; Torralba, Á.; and Hoffmann, J. 2022. Operator-Potential Heuristics for Symbolic Search. In *Proc. AAAI'22*. Accepted.

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

Franco, S.; Torralba, A.; Lelis, L. H.; and Barley, M. 2017. On Creating Complementary Pattern Databases. In *Proc. IJCAI'17*, 4302–4309.

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.

Haslum, P.; and Geffner, H. 2000. Admissible Heuristics for Optimal Planning. In *Proc. AIPS'00*, 140–149.

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.

Jensen, R. M.; Veloso, M. M.; and Bryant, R. E. 2008. State-set branching: Leveraging BDDs for heuristic search. *Artificial Intelligence*, 172(2-3): 103–139.

Kissmann, P.; and Edelkamp, S. 2011. Improving Cost-Optimal Domain-Independent Symbolic Planning. In *Proc. AAAI'11*, 992–997.

Massey, B. 1999. *Directions in planning: Understanding the flow of time in planning*. Ph.D. thesis, University of Oregon.

McMillan, K. L. 1993. *Symbolic Model Checking*. Kluwer Academic Publishers.

Pettersson, M. P. 2005. Reversed planning graphs for relevance heuristics in AI planning. In *Planning, Scheduling and Constraint Satisfaction: From Theory to Practice*, volume 117, 29–38. IOS Press. ISBN 978-1-58603-484-9.

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. 2015. From Non-Negative to General Operator Cost Partitioning. In *Proc. AAAI'15*, 3335–3341.

Seipp, J. 2018. Fast Downward Scorpion. In *IPC 2018 planner abstracts*, 77–79.

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.; and Helmert, M. 2015. New Optimization Functions for Potential Heuristics. In *Proc. ICAPS'15*, 193–201.

Sievers, S.; Wehrle, M.; and Helmert, M. 2016. An Analysis of Merge Strategies for Merge-and-Shrink Heuristics. In *Proc. ICAPS'16*, 294–298.

Speck, D.; Geißer, F.; and Mattmüller, R. 2020. When Perfect Is Not Good Enough: On the Search Behaviour of Symbolic Heuristic Search. In *Proc. ICAPS'20*, 263–271.

Suda, M. 2013. Duality in STRIPS planning. *CoRR*, abs/1304.0897.

Torralba, Á.; Alcázar, V.; Kissmann, P.; and Edelkamp, S. 2017. Efficient symbolic search for cost-optimal planning. *Artificial Intelligence*, 242: 52–79.

Torralba, Á.; López, C. L.; and Borrajo, D. 2018. Symbolic perimeter abstraction heuristics for cost-optimal planning. *Artificial Intelligence*, 259: 1–31.