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

Admissible Heuristics for Multi-Objective Planning

Florian Geißer, Patrik Haslum, Sylvie Thiébaux, Felipe Trevizan

The Australian National University, School of Computing
florian.geisser.work@gmail.com,
{patrik.haslum,sylvie.thiebaux,felipe.trevizan}@anu.edu.au

Abstract

Planning problems of practical relevance commonly include multiple objectives that are difficult to weight a priori. Several heuristic search algorithms computing the Pareto front of non-dominated solutions have been proposed to handle these multi-objective (MO) planning problems. However, the design of informative admissible heuristics to guide these algorithms has not received the same level of attention. The standard practice is to use the so-called ideal point combination, which applies a single-objective heuristic to each objective independently, without capturing any of the trade-offs between them. This paper fills this gap: we extend several classes of classical planning heuristics to the multi-objective case, in such a way as to reflect the tradeoffs underlying the various objectives. We find that MO abstraction heuristics achieve overall the best performance, but that not every MO generalisation pays off.

1 Introduction

Many optimisation problems, including planning, in practice have multiple, sometimes competing objectives (e.g., cost, risk, efficiency, and fairness), and often users cannot quantify a priori exactly how they desire to balance these. Multi-objective (MO) optimisation (Ehrgott 2005; Roijers et al. 2013) tackles this by generating a complete set of non-dominated options, known as the Pareto front, which represents the possible trade-offs between the objectives, instead of a single optimal solution.

Although several algorithms for MO heuristic state-space search have been proposed (Stewart and White 1991; Mandow and Pérez-de-la-Cruz 2010; Ulloa et al. 2020; Ahmadi et al. 2021), no attention has been paid to the question of how to derive informative admissible MO heuristics. In fact, all previous works we are aware of have used the very simple, so-called ideal point heuristic, which applies a single-objective admissible heuristic to each objective independently. While this method has the advantage of being general, in the sense that any admissible heuristic for each objective can be used, it results in a MO heuristic that fails to capture necessary trade-offs between objectives, and therefore offers little or no utility over blind search for problems with a non-trivial objective combination.

To illustrate, consider a simple project planning domain, with N tasks to be completed. For each task, the options are to do it by ourselves (taking time, but no cost) or to outsource (costing money, but taking none of our time). The two objectives are (own) time and money spent. From any state, it is possible to solve the problem with either zero time or zero money spent but of course not both. Therefore, the ideal point combination of any two single-objective heuristics, including, $h^\star$, will be a zero vector, and no better than blind search. In contrast, a genuine MO heuristic will be a set of cost vectors that collectively lower-bound the costs of all non-dominated plans.

MO heuristic search algorithms, such as NAMOA* (Mandow and Pérez-de-la-Cruz 2010), are specifically designed to accommodate these genuine MO heuristics, in the form of a set of cost vectors capable of capturing trade-offs between objectives. For such a heuristic to be admissible, each element of the Pareto front must be dominated (or equalled) by at least one cost vector in the heuristic set.

This paper addresses the question of how to construct these more powerful admissible heuristics for MO domain-independent planning. In particular, we find several classes of well-known admissible planning heuristics, namely abstraction-based, critical path-based, and LP/IP-based heuristics, whose computation can be extended to the multi-objective case in a principled fashion. We also propose MO analogues of heuristic combination by maximisation and cost partitioning. We prove these new MO heuristics are admissible. We compare their informativeness, run-time and coverage, relative to each other and their ideal point counterparts, using an improved version of NAMOA*. It turns out that MO abstraction heuristics outperform their ideal point counterpart, and have overall the strongest performance on the benchmark set we consider. The same however, does not generally hold true for all other MO heuristics.

We also briefly discuss ways of compiling MO planning into classical or numeric planning, though we did not find any of these compilations to be practical. We conclude there are several challenging open questions in both theory and implementation of heuristic search for MO planning.

2 Background

We consider planning problems represented in the STRIPS formalism (Fikes and Nilsson 1971) but where actions in-

duce costs for multiple objectives simultaneously. Formally, a STRIPS planning task is a tuple $\mathcal{T} = \langle P, A, s_I, G \rangle$, where P is a finite set of propositions and A is a finite set of actions. A *state* $s \subseteq P$ is a set of propositions and describes a situation of the world where exactly those propositions are true. The initial state is given by $s_I \subseteq P$, whereas the goal $G \subseteq P$ is a set of propositions we want to achieve. A state s is called a goal state iff $G \subseteq s$. Each action $a \in A$ is a tuple $a = \langle \mathsf{pre}(a), \mathsf{add}(a), \mathsf{del}(a) \rangle$, where $\mathsf{pre}(a) \subseteq P$ is the set of propositions that must be true in order for a to be applicable, i.e. a is applicable in state s iff $\mathsf{pre}(a) \subseteq s$. Once a is applied in s its add-effects $\mathsf{add}(a) \subseteq P$ and delete-effects $\mathsf{del}(a) \subseteq P$ are applied, resulting in the *successor state* $s[a] = (s \setminus \mathsf{del}(a)) \cup \mathsf{add}(a)$. A sequence of actions $\pi = \langle a_1, \ldots, a_n \rangle$ is applicable in state s if a_i is applicable in $s[a_1] \ldots [a_{i-1}]$ for all i, and we denote the successor state as $s[\pi]$. A solution to the planning task $\mathcal{T}$, a *plan*, is a sequence of actions $\pi = \langle a_1, \ldots, a_n \rangle$ that is applicable in s_I and leads to a goal state, i.e. $G \subseteq s_I[\pi]$.

2.1 Multi-objective Planning

A *multi-objective* planning (MOP) task is a STRIPS planning task with k cost functions, $c_1, \ldots, c_k$. Each action a is associated with a k-dimensional cost vector, $\vec{c}(a) \in \mathbb{N}^k$, where the ith component, denoted $\vec{c}(a)^i$, is a's contribution to c_i. Note that we assume each action's cost, for each cost function, is a non-negative integer. The cost of a sequence of actions $\pi = \langle a_1, \ldots, a_n \rangle$ is also a vector, $\vec{c}(\pi) = \sum_{i=1}^{n} \vec{c}(a_i)$, where the sum is taken with vector addition. With slight abuse of notation, we apply $\vec{c}$ also to sets of plans: if Π is a set of plans, then $\vec{c}(\Pi) = \{\vec{c}(\pi) \mid \pi \in \Pi\}$ is the set of cost vectors of plans in Π. Given two cost vectors $\vec{v}_1, \vec{v}_2$, we say that $\vec{v}_1$ *Pareto dominates* $\vec{v}_2$, denoted $\vec{v}_1 \prec \vec{v}_2$, iff for all $i = 1 \ldots k : \vec{v}_1^i \leq \vec{v}_2^i$ and additionally $\vec{v}_1 \neq \vec{v}_2$. Note that $\vec{v}_1 \prec \vec{v}_2$ implies $\vec{v}_1^i < \vec{v}_2^i$ for at least one i. Dominance is a strict partial order, i.e., it is transitive and asymmetric. We say $\vec{v}_1$ *dominates or equals* $\vec{v}_2$, denoted $\vec{v}_1 \preceq \vec{v}_2$, iff $\vec{v}_1 \prec \vec{v}_2$ or $\vec{v}_1 = \vec{v}_2$. A plan π_1 dominates (resp. dominates or equals) plan π_2 if $\vec{c}(\pi_1) \prec \vec{c}(\pi_2)$ (resp. $\vec{c}(\pi_1) \preceq \vec{c}(\pi_2)$).

A MOP task can have multiple solution plans whose cost vectors are mutually non-dominating, and not equal; these represent different possible trade-offs between the k cost functions. The problem of MOP can be defined as computing the set of all non-dominated plans, known as the *Pareto front*, or at least one representative of each such trade-off, known as a *Pareto coverage set*. We focus on the latter.

Formally, let $\mathcal{T}$ be a MOP task and $\Pi(\mathcal{T})$ the set of all plans for $\mathcal{T}$. The *Pareto front* $\mathsf{PF}(\mathcal{T})$ is the set of plans which are not dominated by any other plan in Π. Note that the Pareto front contains *all* possible non-dominated plans, which can include multiple plans with equal cost vectors. In the presence of (cycles of) $\vec{0}$-cost actions, the Pareto front may even be infinite. A *Pareto coverage set* $\mathsf{PCS}(\mathcal{T})$ is a subset of $\mathsf{PF}(\mathcal{T})$ such that for each plan $\pi' \in \Pi(\mathcal{T})$, $\mathsf{PCS}(\mathcal{T})$ contains a plan that dominates or equals π'. Formally, $\mathsf{PCS}(\mathcal{T}) \subseteq \mathsf{PF}(\mathcal{T})$ and $\forall \pi' \in \Pi(\mathcal{T}) \; \exists \pi \in \mathsf{PCS}(\mathcal{T}) \; \vec{c}(\pi) \preceq \vec{c}(\pi')$.

The Pareto front of a task is unique, but the Pareto coverage set is not, since we may choose different representatives for each non-dominated cost vector. Note, however, that $\vec{c}(\mathsf{PCS}(\mathcal{T}))$ is unique (i.e., whatever representatives are chosen they collectively have the same set of cost vectors) and $\vec{c}(\mathsf{PCS}(\mathcal{T})) = \vec{c}(\mathsf{PF}(\mathcal{T}))$. In the special case that $k = 1$, computing the PCS reduces to finding a plan with minimum cost, i.e., classical optimal planning.

With slight abuse of notation, we write $\mathsf{PF}(\mathcal{T}, s', G')$ and $\mathsf{PCS}(\mathcal{T}, s', G')$ for the Pareto front and coverage set of plans starting from a state s' different from s_I and/or with a goal G' different from G. We usually omit $\mathcal{T}$, as it is clear from context, and may omit s' or G' when they are unchanged.

2.2 Multi-objective Heuristic Search

In multi-objective state-space search, the heuristic value of a state is not a single number, but a set of cost vectors. That is, $H(s) \subset \mathbb{N}^k$. (Note: We will use H for MO heuristics and h for single-objective heuristics.) A MO heuristic is admissible iff for every state and for every plan from the state, this set contains a cost vector that dominates or equals that of the plan (Mandow and Pérez-de-la-Cruz 2010). Formally:

Definition 1. *A MO heuristic H is* admissible *iff* $\forall s \in \mathcal{S}(\mathcal{T}) \; \forall \pi \in \mathsf{PF}(s) \; \exists \vec{v} \in H(s) \; \vec{v} \preceq \vec{c}(\pi)$.

This notion of admissibility implies that the perfect heuristic equals the set of costs of plans in any Pareto coverage set, i.e., $H^\star(s) = \vec{c}(\mathsf{PCS}(s))$. For single-objective search problems it is well known that the A* algorithm together with an admissible heuristic yields an optimal solution. A similar result exists for multi-objective search: the NAMOA* best-first search algorithm (Mandow and Pérez-de-la-Cruz 2010) returns an optimal solution if the underlying heuristic is admissible.

2.3 The Ideal Point Heuristic

Experimental evaluations of MO heuristic search algorithms (Stewart and White 1991; Machuca et al. 2009; Ulloa et al. 2020) all use the simple ideal point heuristic which estimates each objective independently.

Formally, given admissible heuristics $h_1, \ldots, h_k$ for each of the k objectives, the ideal point heuristic combines them to form a cost vector with the value of each heuristic, i.e., $\vec{v}'(s) = \langle h_1(s), \ldots, h_k(s) \rangle$. $\vec{v}'(s)$ dominates or equals the cost of every solution in the Pareto front, because each h_i lower-bounds the minimum value of the ith objective in any solution plan. Hence, the single-vector set $H_{\text{ideal}}(s) = \{\vec{v}'(s)\}$ is an admissible MO heuristic.

A strength of the ideal point heuristic is that it combines single-objective "black box" heuristics in an admissible way, without altering their computation. As an MO heuristic, however, it also has an obvious weakness, in that it fails to account for any trade-off between the different objectives, as illustrated by our introductory example.

3 Heuristics for MO Planning

We first describe the multi-objective analogues of two important general operations: taking the maximum, and admissible sum via cost partitioning, of admissible MO heuris-

tics. We then discuss, in turn, our generalisation of abstraction heuristics, critical path heuristics and operator counting heuristics based on (integer) linear programming.

3.1 Maximisation of MO Heuristics

In single-objective heuristic search, taking the maximum, per state, of two or more admissible heuristics yields an admissible heuristic. This is possible also with MO heuristics, provided an appropriate definition of "maximum". However, what is the maximum of two sets of cost vectors, which are only partially ordered by dominance, is neither obvious nor unique. We propose two such definitions: both preserve admissibility, but they differ on some other properties. In particular, one preserves heuristic consistency, but the other does not. This turns out to have a significant impact on the NAMOA* algorithm. We state both definitions before proceeding to prove their properties.

Definition 2. *Let V_1 and V_2 be sets of cost vectors of dimension k. The* component-wise multi-objective maximum *of V_1 and V_2 is* $\operatorname{comax}(V_1, V_2) = \{\langle \max(\vec{v}_1^1, \vec{v}_2^1), \max(\vec{v}_1^2, \vec{v}_2^2), \ldots, \max(\vec{v}_1^k, \vec{v}_2^k)\rangle \mid \vec{v}_1 \in V_1, \vec{v}_2 \in V_2\}$.

That is, $\operatorname{comax}(V_1, V_2)$ contains one vector for each pair of vectors from V_1 and V_2, respectively, which is their component-wise maximum. Hence, in the worst case $|\operatorname{comax}(V_1, V_2)| = |V_1||V_2|$.

Let V be a set of cost vectors. We write $\mathsf{ND}(V) = \{\vec{v} \in V \mid \vec{v}' \not\prec \vec{v}\, \forall \vec{v}' \in V\}$ for the set of non-dominated vectors in V.

Definition 3. *Let V_1 and V_2 be sets of k-dimensional cost vectors. The* anti-dominance maximum *of V_1 and V_2 is* $\operatorname{admax}(V_1, V_2) = \{\vec{v_1} \in \mathsf{ND}(V_1) \mid \forall \vec{v_2} \in \mathsf{ND}(V_2) : \vec{v_1} \not\prec \vec{v_2}\} \cup \{\vec{v_2} \in \mathsf{ND}(V_2) \mid \forall \vec{v_1} \in \mathsf{ND}(V_1) : \vec{v_2} \not\prec \vec{v_1}\}$.

Note that $|\operatorname{admax}(V_1, V_2)| \leq |V_1| + |V_2|$, since it contains only elements of the two sets.

Both definitions have some natural properties: they are both commutative and associative, so the maximum of more than two sets of vectors is unique; and the maximum of two sets of vectors is dominated by both sets. Most important, however, is the following:

Proposition 4. *Let H_1 and H_2 be admissible MO heuristics:*
(i) $H(s) = \operatorname{comax}(H_1(s), H_2(s))$ is admissible;
(ii) $H(s) = \operatorname{admax}(H_1(s), H_2(s))$ is admissible.

Proof. Let π_s be a non-dominated plan from state s. By admissibility, there exists $\vec{v}_1 \in H_1(s)$ and $\vec{v}_2 \in H_2(s)$ such that $\vec{v}_1 \preceq \vec{c}(\pi_s)$ and $\vec{v}_2 \preceq \vec{c}(\pi_s)$. W.l.o.g., we can assume $\vec{v}_1 \in \mathsf{ND}(H_1(s))$ and $\vec{v}_2 \in \mathsf{ND}(H_2(s))$, since if they are dominated by some other vector in the respective set, we can just pick the dominating vector instead; by transitivity, it will also dominate $\vec{c}(\pi_s)$.

(i) By definition, there is a vector $\vec{u} = \langle \max(\vec{v}_1^1, \vec{v}_2^1), \ldots, \max(\vec{v}_1^k, \vec{v}_2^k)\rangle$ in $\operatorname{comax}(V_1, V_2)$. Consider the ith objective: Since $\vec{v}_1 \preceq \vec{c}(\pi_s)$, we have $\vec{v}_1^i \leq \vec{c}(\pi_s)^i$, and likewise since $\vec{v}_2 \preceq \vec{c}(\pi_s)$, we have $\vec{v}_2^i \leq \vec{c}(\pi_s)^i$. Thus, $\vec{u}^i = \max(\vec{v}_1^i, \vec{v}_2^i) \leq \vec{c}(\pi_s)^i$. Since this holds for all k objectives, $\vec{u} \preceq \vec{c}(\pi_s)$.

(ii) At least one of $\vec{v}_1$ and $\vec{v}_2$ is in $\operatorname{admax}(H_1(s), H_2(s))$, since it cannot be the case that both $\vec{v}_1 \prec \vec{v}_2$ and $\vec{v}_2 \prec \vec{v}_1$. □

Mandow and Pérez-de-la-Cruz (2010) define consistency of an MO heuristic as follows: H is consistent iff for all states s, t and every non-dominated s-t-path P, $\forall \vec{v} \in H(t)\, \exists \vec{u} \in H(s)\, (\vec{u} \preceq \vec{c}(P) + \vec{v})$ holds. Like in the single-objective case, consistency holds iff it holds for all transitions (i.e., paths of length 1). They show that NAMOA*, like A*, has the property that if the heuristic is consistent then no search node once closed will need to be reopened. comax preserves consistency (Proposition 5), but admax does not. This is important because we confirm experimentally that MO heuristics that are inconsistent cause NAMOA* to perform a sometimes significant number of re-expansions.

Proposition 5. *Let H_1 and H_2 be consistent MO heuristics. Then $H(s) = \operatorname{comax}(H_1(s), H_2(s))$ is consistent.*

Proof. Consider a transition from state s to state t with cost $\vec{c}$. To prove consistency we need to prove that for every $\vec{v} \in H(t)$ there exists some $\vec{u} \in H(s)$ such that $\vec{u} \preceq \vec{c} + \vec{v}$. Since $H(t) = \operatorname{comax}(H_1(t), H_2(t))$, $\vec{v}$ is the coordinate-wise maximum of two vectors $\vec{v}_1 \in H_1(t)$ and $\vec{v}_2 \in H_2(t)$. Because H_1 and H_2 are consistent, there exist $\vec{u}_1 \in H_1(s)$ such that $\vec{u}_1 \preceq \vec{c} + \vec{v}_1$ and $\vec{u}_2 \in H_2(s)$ such that $\vec{u}_2 \preceq \vec{c} + \vec{v}_2$. Let $\vec{u}'$ be the coordinate-wise maximum of $\vec{u}_1$ and $\vec{u}_2$. By definition, $\vec{u}'$ is part of $H(s) = \operatorname{comax}(H_1(s), H_2(s))$. Thus, it suffices to prove that $\vec{u}' \preceq \vec{c} + \vec{v}$. Consider the ith objective: $\vec{v}^i = \max(\vec{v}_1^i, \vec{v}_2^i)$. From consistency of H_1 and H_2, respectively, we have $\vec{u}_1^i \leq \vec{c}^i + \vec{v}_1^i$, and $\vec{u}_2^i \leq \vec{c}^i + \vec{v}_2^i$, and hence $\max(\vec{u}_1^i, \vec{u}_2^i) \leq \vec{c}^i + \max(\vec{v}_1^i, \vec{v}_2^i)$. As this holds for all objectives, $\vec{u}' \preceq \vec{c} + \vec{v}$. □

In the remainder of the paper, we will use momax to refer to any multi-objective maximum operator that preserves heuristic admissibility (such as comax or admax).

3.2 Additivity and Cost Partitioning

Another way to strengthen admissible heuristics is to exploit conditions under which the values of two or more heuristics can be admissibly added together. A general method to ensure this is via cost partitioning (Katz and Domshlak 2008; Haslum, Bonet, and Geffner 2005). This can be done also in the MO case. First, we need to define how sets of cost vectors are added:

Definition 6. *Let $V_1, \ldots, V_n$ be sets of k-dimensional cost vectors (i.e., $V_i \subset \mathbb{N}^k$). The* multi-objective sum *is $V_1 + \ldots + V_n = \{\vec{v}_1 + \ldots + \vec{v}_n \mid \vec{v}_1 \in V_1, \ldots, \vec{v}_n \in V_n\}$, i.e., the set of all sums of one vector from each set.*

Definition 7. *A* cost partitioning *for an MOP task $\mathcal{T}$ with k-dimensional action cost function $\vec{c}$ is a collection of functions $\vec{c}_1, \ldots, \vec{c}_n$, each mapping actions to $\mathbb{N}^k$, such that $\vec{c}_1(a) + \ldots + \vec{c}_n(a) \preceq \vec{c}(a)$ for each action a in $\mathcal{T}$.*

Let $H/\vec{c}_i$ denote the MO heuristic H computed using $\vec{c}_i$ in place of the original cost function $\vec{c}$. To apply a cost partitioning, we compute $H/\vec{c}_i$ for each partition $\vec{c}_i$ and add the results. Note that the same or a different admissible heuristic may be used for each partition.

Proposition 8. *Let $\Sigma = \{\vec{c}_1, \ldots, \vec{c}_n\}$ be a cost partitioning and $H_1, \ldots, H_n$ admissible MO heuristics. Then $H_\Sigma(s) = \sum_{i=1,\ldots,n} H_i/\vec{c}_i(s)$ is admissible.*

Proof. Let π be a non-dominated plan from state s. Because the cost of a plan is the sum of the cost vectors of actions in it, and by definition of a cost partitioning, $(\vec{c}_1(\pi) + \ldots + \vec{c}_n(\pi)) \preceq \vec{c}(\pi)$. Since $H_i/\vec{c}_i$ is admissible w.r.t. $\vec{c}_i$, there is a vector $\vec{v}_i \in H_i/\vec{c}_i(s)$ such that $\vec{v}_i \preceq \vec{c}_i(\pi)$, for $i = 1, \ldots, n$. Hence, $(\vec{v}_1 + \ldots + \vec{v}_n) \preceq \vec{c}(\pi)$, and $(\vec{v}_1 + \ldots + \vec{v}_n) \in H_\Sigma(s)$. □

An important special case is *disjoint* cost partitionings, which for each action assign its full cost vector to one of the partitions (and $\vec{0}$ to every other). We say that heuristics $H_1, \ldots, H_n$ are *additive* iff there is a disjoint cost partitioning $\vec{c}_i, \ldots, \vec{c}_n$ such that $H_i/\vec{c}_i$ equals H_i for all states, i.e., each of the heuristics is unchanged by not counting the cost of actions not in its partition. This means additive heuristics can be admissibly added without modifying their computation. Projection abstractions, which we discuss in the next section, give rise to additive heuristics under certain conditions. However, more general cost partitionings have been shown to result in stronger heuristics for single-objective planning (e.g., Seipp, Keller, and Helmert 2017).

3.3 Abstraction Heuristics

Abstraction heuristics use a mapping φ from the state space of the problem into a, usually smaller, abstract state space; if the mapping is homomorphic (preserves existence and cost of state transitions), then optimal path cost in the abstract space is an admissible heuristic for search in the original space, i.e., $h^\varphi(s) = h^\star(\varphi(s))$. If the abstract space is small enough, $h^\star(\varphi(s))$ can be computed by blind search. This is efficient because many states map to the same abstract state, but the cost is (pre-)computed only once, stored in memory, and retrieved by a fast lookup for state evaluation. In planning, the abstraction function is usually defined on the planning task and yields a corresponding abstract task, $\mathcal{T}^\varphi$, whose induced state space is the abstract space. Well known examples of abstraction heuristics in planning are pattern databases (PDBs) (Edelkamp 2001), based on projection mappings, and the merge-and-shrink heuristic (Helmert et al. 2014).

Extending the principle of abstraction heuristics to the MO setting is straightforward: instead of computing a single optimal path cost for each abstract state, we compute a Pareto coverage set, i.e., $H^\varphi(s) = \vec{c}(\mathsf{PCS}(\mathcal{T}^\varphi, \varphi(s)))$.

Proposition 9. *H^φ is admissible if the abstraction mapping preserves state transitions with equal or dominating cost vectors, i.e., for any pair of states s, s' in the original state space, if there is an action a such that $s[a] = s'$, then there is an abstract action a^φ such that $\varphi(s)[a^\varphi] = \varphi(s')$ and $\vec{c}(a^\varphi) \preceq \vec{c}(a)$.*

Proof. Let $\pi_s = \langle a_1, \ldots, a_n \rangle$ be a non-dominated plan from a state s, with cost $\vec{c}(\pi_s)$. Because φ preserves state transition cost vectors, there exists a corresponding abstract plan $\pi'_{\varphi(s)} = \langle a'_1, \ldots, a'_n \rangle$, whose cost equals or dominates $\vec{c}(\pi_s)$. If $\pi'_{\varphi(s)}$ is non-dominated in the abstract task, then $\vec{c}(\pi'_{\varphi(s)}) \in \vec{c}(\mathsf{PCS}(\mathcal{T}^\varphi, \varphi(s)))$; otherwise there is another plan $\pi''_{\varphi(s)}$ from $\varphi(s)$ such that $\vec{c}(\pi''_{\varphi(s)}) \prec \vec{c}(\pi'_{\varphi(s)})$ and $\vec{c}(\pi''_{\varphi(s)}) \in \vec{c}(\mathsf{PCS}(\mathcal{T}^\varphi, \varphi(s)))$. In either case, there exists $\vec{v} \in H^\varphi(s)$ such that $\vec{v} \preceq \vec{c}(\pi_s)$. □

Abstraction functions commonly used for planning tasks, such as projection on a subset of state variables and variable domain projections, satisfy this condition because each action in the original task is represented in the abstract task. Efficient implementations of abstraction heuristics compute $h^\star$ for all abstract states by a single backward search from the abstract goal, using an algorithm like Dijkstra's (Edelkamp 2001; Sievers, Ortlieb, and Helmert 2012). Similar algorithms exist for MO shortest-path (e.g., Ehrgott 2005).

Abstraction heuristics are additive, as defined in the previous section, when the sets of actions that have a non-empty effect in the corresponding abstract spaces are disjoint (Edelkamp 2001). This is true also in the MO case. The single-objective canonical heuristic (Haslum et al. 2007) is an abstraction heuristic based on PDBs that makes use of disjoint additivity. Let C be a collection of abstractions and $\mathcal{A}$ the collection of all maximal (w.r.t. set inclusion) additive subsets of C. The canonical heuristic function of C is $h^C(s) = \max_{\Phi \in \mathcal{A}} \sum_{\varphi \in \Phi} h^\varphi(s)$. Its extension to the MOP case is now straightforward: $H^C(s) = \mathrm{momax}_{\Phi \in \mathcal{A}}(\sum_{\varphi \in \Phi} H^\varphi(s))$, where the sum over each subset Φ is the MO sum (Definition 6).

To generate a collection of abstractions to be combined in this way, algorithms from classical planning can be used as-is if they do not depend on the heuristic values; examples include systematic enumaration, used by Pommerening et al. (2013), and Edelkamp's (2001) bin-packing method.

3.4 Critical Path Heuristics

Critical path heuristics (Haslum and Geffner 2000) is the family of heuristics denoted by h^m ($m \in \mathbb{N}_{>0}$) that relaxes the cost of achieving a set propositions $\Gamma \subseteq P$ by that of achieving the most expensive subset of Γ of size m. They are defined as follows: Let $R(\Gamma)$ be the set of pairs $\langle \Gamma', a \rangle$, where Γ is a set of propositions and a an action, such that $\Gamma \cap \mathsf{add}(a) \neq \emptyset$, $\Gamma \cap \mathsf{del}(a) = \emptyset$, and $\Gamma' = (\Gamma \setminus add(a)) \cup \mathsf{pre}(a)$. In each pair, Γ' is the regression of Γ through action a. Given a state $s \subseteq P$, $h^m(s) = g^m_s(G)$, where

$$g^m_s(\Gamma) = \begin{cases} 0 & \text{if } \Gamma \subseteq s, \\ \min\limits_{\langle \Gamma', a \rangle \in R(\Gamma)} c(a) + g^m_s(\Gamma') & \text{if } |\Gamma| \leq m, \Gamma \not\subseteq s, \\ \max\limits_{\Gamma' \subset \Gamma, |\Gamma'| = m} g^m_s(\Gamma') & \text{otherwise.} \end{cases}$$

This set of recursive equations implicitly solves a shortest-path problem from s to each Γ while applying the relaxation to sets Γ of size greater than m. The h^m family of heuristics is polynomial in $|P|$ and $|A|$ for a fixed m, while being exponential in m. A notable special case is h^1 which is equivalent to $h^{\max}$ (Bonet and Geffner 1999).

In the MO setting, $g^m_s(\Gamma)$ is a set of non-dominated cost vectors. In the base case (when $\Gamma \subseteq s$), it is $\{\vec{0}\}$. The $\min$

over achievers in the second case is replaced by taking the non-dominated vectors from the union of sets given by each action, i.e., $\mathsf{ND}(\bigcup_{\langle \Gamma', a\rangle \in R(\Gamma)} \{\vec{c}(a) + \vec{v} \mid \vec{v} \in g_s^m(\Gamma')\})$, adding the vector $\vec{c}(a)$ in place of the single action cost. The $\max$ in the last case is replaced by any admissibility-preserving momax operator, as defined in Section 3.1.

This MO-h^m heuristic is admissible, and can be computed by the same label-correcting procedure used in the single-objective case. Depending on the momax operator used, the heuristic may or may not be consistent.

3.5 IP/LP-based Heuristics

Recently, several admissible classical planning heuristics based on (Integer) Linear Programming (IP/LP) formulations have been proposed. Constraints of the IP/LP express properties that must be satisfied by any valid plan (from the evaluated state), and the objective represents the plan cost. Pommerening et al. (2014) introduced the operator counting framework, which allows several IP/LP-based heuristics to be expressed in a common form and combined. The general form of an operator counting problem is

$$\min \sum_{a \in A} c(a) Y_a \text{ s.t. } C(Y_a : a \in A)$$

where $c(a)$ is the cost of action a, Y_a represents the number of occurrences of a in a plan, and $C(Y_a : a \in A)$ is a set of operator count constraints, which lower-bound the counting variables. Counting constraints can be formulated from several kinds of information derived from the planning task, including landmarks (Karpas and Domshlak 2009), net change equations, and abstractions. Several types of constraints can be combined as long as they share only the counting variables. Although in theory the counting variables are integers, in practice the quicker-to-solve LP relaxation (allowing fractional counts) is used. Imai and Fukunaga (2015) proposed an IP formulation of the optimal delete relaxation heuristic, h^+, which also fits the framework.

In principle, extending operator counting heuristics to MO planning is simple: we replace the scalar $c(a)$ with the vector $\vec{c}(a)$, resulting in the multi-objective IP/LP,

$$\min \sum_{a \in A} \vec{c}(a) Y_a \text{ s.t. } C(Y_a : a \in A) \tag{1}$$

and compute its PCS. Because the counting constraints enforce lower bounds on the counting variables that hold for all valid plans (from state s), this is admissible. In practice, however, computing the PCS of a multi-objective LP is more complex than solving a single-objective LP.

Multi-objective Linear Programming The Pareto front of a multi-objective linear program (MOLP) can be computed in several ways, e.g., by a simplex-like algorithm or by weighted sum methods (e.g., Ehrgott 2005). Like an ordinary LP, the feasible set of an MOLP, and its image in the objective space, are convex polyhedra. However, non-dominated solutions do not necessarily lie on the polyhedron's vertices; entire facets can represent an infinite set of different non-dominated solutions. As an example, consider a simple MOLP with objectives $c_1 = x$, $c_2 = y$ and constraints $x + y \geq 1$, $x, y \geq 0$: every single point on the line $x + y = 1$, between $x = 1, y = 0$ and $x = 0, y = 1$, is a non-dominated solution, representing a different trade-off between c_1 and c_2. This causes a problem for MO heuristic search algorithms since these enumerate the elements of the heuristic set.

An admissible discrete heuristic set can be obtained by taking all mutually non-dominated integral cost vectors that are on or dominated by the Pareto front of the MOLP (see Figure 1). For the Pareto front of an operator counting MOLP this set is finite, and it is the strongest discretisation that is admissible. This is analogous to how in classical optimal planning, the value obtained from an LP-based heuristic can admissibly be rounded up to the nearest integer, since action costs are integral.

Iterative Method As an alternative, we propose a method which solves a series of IPs. It repeatedly finds a new element of $H(s)$ by adding constraints to ensure that the cost of the new solution is not dominated by any element already found. The process terminates when no such vector exists.

Formally, let $C(Y_a : a \in A)$ be a set of counting constraints that are valid for plans from state s, and initialise $V = \emptyset$. Define the non-dominance constraint as

$$\mathrm{ndc}(V) = \bigwedge_{\vec{v} \in V} \left(\bigvee_{i=1,\ldots,k} \left(\sum_{a \in A} c_i(a) Y_a \leq \vec{v}^i - 1 \right) \right).$$

We can now solve

$$\min f(\ldots) : C(Y_a : a \in A) \cup \mathrm{ndc}(V) \tag{2}$$

to obtain a vector of action counts $\vec{y} = \langle y_a : a \in A \rangle$, and compute the corresponding cost vector $\vec{v} = \sum_{a \in A} \vec{c}(a) y_a$. Update $V = V \cup \{\vec{v}\}$ and repeat until the constrained problem (2) is infeasible. Then $H(s) = V$.

The objective $f(\ldots)$ in (2) can be chosen freely, as long as it ensures that the cost vector $\vec{v}$ found is non-dominated for the MO problem (1). One way to ensure this is to use a weighted sum of the cost functions, with strictly positive weights (Proposition 10 below), but other methods are also possible (cf., e.g., Ehrgott 2005).

The disjunction in the non-dominance constraint means that (2) is not an LP when $V \neq \emptyset$. It can be solved as an IP, using indicator variables (which are supported by most modern IP solvers) or a big-M formulation. The MO heuristic computed by the iterative method is admissible, provided the cost vectors of all non-dominated solutions to (1) are integral (Proposition 12 below). This integrality constraint is necessary, as shown by the example in Figure 1: vectors $\vec{v}_1$, $\vec{v}_2$ and $\vec{v}_3$ are integer cost vectors, that may correspond to plan costs. $\vec{u}$ is a non-dominated solution to the MOLP. If $\vec{u}$ is added to V first, then neither $\vec{v}_1$ nor $\vec{v}_3$ will be found, since neither is less than $\vec{u}$ by 1 in at least one dimension.

A simple way to ensure integer cost vectors is to constrain the counting variables to be integer, i.e., to solve (1) as an MOIP rather than its MOLP relaxation. This does however come at a computational cost.

In the remainder of this section, we prove termination and admissibility of the iterative method.

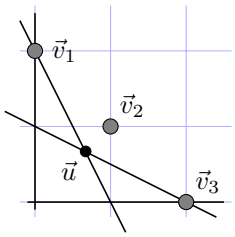

Figure 1: The Pareto front of the MOLP consists of the line segments $\vec{v}_1$–$\vec{u}$ and $\vec{u}$–$\vec{v}_3$. The set $\{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$ is the admissible discretisation of the Pareto front.

Proposition 10. *The cost vector corresponding to an optimal solution to (2) with $f = \sum_{i=1,\dots,k} w_i \left(\sum_{a \in A} c_i(a) Y_a\right)$ and $w_i > 0$ for $i = 1, \dots, k$, is non-dominated.*

Proof. That optimal solutions to the weighted sum problem are non-dominated for an MO problem when weights are strictly positive is well known, and easy to prove (e.g., Ehrgott 2005). The only complication here is the non-dominance constraints in (2), which are not part of (1).

Let $\vec{y} = \langle y_a : a \in A \rangle$ be the optimal solution to (2). Clearly $\vec{y}$ satisfies $C(Y_a : a \in A)$, so it is a solution to (1). It remains to show that it is non-dominated. Let $\vec{v} = \sum_{a \in A} \vec{c}(a) y_a$, and $o = \sum_{i=1,\dots,k} w_i \vec{v}^i$ the value of the weighted sum objective f. Suppose there is another solution $\vec{y}'$ to (1), with cost vector $\vec{v}'$, such that $\vec{v}' \prec \vec{v}$. Then $\vec{v}'^i \leq \vec{v}^i$ for all $i = 1, \dots, k$, and $\vec{v}'^i < \vec{v}^i$ for some i, and thus $o' = \sum_{i=1,\dots,k} \vec{v}'^i < o$. $\vec{y}$ satisfies $\mathrm{ndc}(V)$, which means that for all $\vec{u} \in V$, there is some i such that $\vec{v}^i \leq \vec{u}^i - 1$. Since $\vec{v}'^i \leq \vec{v}^i$ for all $i = 1, \dots, k$, we must also have $\vec{v}'^i \leq \vec{u}^i - 1$, and thus $\vec{y}'$ satisfies $\mathrm{ndc}(V)$. This contradicts that $\vec{y}$ is an optimal solution to (2). □

Proposition 11. *The iterative method applied to an operator counting MOLP (or MOIP) terminates with a finite set V of cost vectors.*

Proof. The value that each cost function c_i can take in any non-dominated solution to an operator counting MOLP (or MOIP) is bounded. Because the counting constraints are satisfied by every valid plan, the sum of the counting variables is no more than maximum length of a non-looping plan, which is bounded by $2^{|P|}$. Thus, the value of c_i is no more than $2^{|P|} \left(\max_{a \in A} \vec{c}(a)^i\right)$. The non-dominance constraint ensures each new cost vector added to V differs from each vector already in the set by at least 1 in at least one dimension. Since both the maximum value and the smallest difference in each dimension are bounded, V cannot contain an infinite set of vectors. □

Proposition 12. *If the cost vectors of all non-dominated solutions to (1) are integral, then $H(s)$ computed by the iterative method is admissible.*

Proof. Let $H(s) = V$ be the set of cost vectors computed by the iterative method for a state s. Since the computation has terminated, $C(Y_a : a \in A) \cup ndc(V)$ must be infeasible. Let π_s be a non-dominated plan from s. Let $\vec{y}$ be the action counts corresponding to π_s, i.e., y_a equals the number of occurrences of a in π_s for each $a \in A$, and $\vec{v} = \sum_{a \in A} \vec{c}(a) y_a = \vec{c}(\pi_s)$ the corresponding cost vector. Since the counting constraints $C(Y_a : a \in A)$ are valid, they are satisfied by all plans from s, including π_s; hence, $\vec{y}$ is a solution to (1). $\vec{y}$ does not satisfy $\mathrm{ndc}(V)$, since if it did it would be a solution to (2). Since $\vec{v}$ and all $\vec{u} \in V$ are integral, this implies $\vec{v}^i \geq \vec{u}^i$ for all i for some $\vec{u} \in V$. But then either $\vec{u}^i = \vec{v}^i$ for all i, meaning $\vec{u} = \vec{v}$, or $\vec{u}^i < \vec{v}^i$ for some i, meaning $\vec{u} \prec \vec{v}$; in either case $\vec{u} \preceq \vec{v}$. □

4 Compilation Schemes

Before we present an empirical evaluation of above heuristics, we discuss three general schemes for solving a MO planning task via a series of single-objective (classical or numeric) planning tasks. Although we did not find these compilations to be practical, we describe them as a starting point for further research. Note that these compilation schemes can also serve to derive an MO heuristic from arbitrary single-objective heuristics, as long as these provide some form of relaxed solution, not only a heuristic value.

Optimal solutions to the single-objective problem formed from an MO problem by taking a weighted sum, $\sum_{i=1,\dots,k} w_i c_i(\pi)$, are non-dominated if all weights are strictly positive (e.g., Ehrgott 2005). The weighted sum problem for MOP is a classical planning problem with a single additive constant action cost function; it can be handled by any classical cost-optimal planner. However, weighted sum minimisation can only yield solutions whose cost vectors lie on the convex hull of the Pareto front, and thus is complete only for tasks with convex Pareto fronts.

The iterative method described above is in fact a general compilation scheme for MO planning: we can compile MO planning into a series of numeric planning tasks, using fluents $f_1, \dots, f_k$ that track the value of each cost function and the non-dominance constraint added to the goal. This scheme has two disadvantages: First, it reduces to numeric, not classical, planning. The goal disjunctions can be compiled away by standard means (cf., e.g., Nebel 2000), leaving a task with so-called simple numeric conditions (Scala, Haslum, and Thiébaux 2016), but numeric planners still cannot match classical planners for efficiency. Second, to complete the computation of the Pareto cover set, the last task in the series must be proven unsolvable.

An alternative to numeric constraints is to exclude the specific plans (action sequences) already found, rather than their cost vectors. This can be done by modification of the domain and problem (see, e.g., Grastien, Benn, and Thiébaux 2021), remaining in the classical setting. Disadvantages of this method are: First, the number of different plans with the same cost vector can be (exponentially) large. Second, the encoding we tried requires a large number of conditional effects, and is intractable even with very few plans excluded.

5 Empirical Evaluation

Evaluations of MO heuristic search algorithms have used bi-objective path-finding on random grids (e.g., Machuca et al.

2009, 2012) or road maps from the 9th DIMACS shortest path challenge (e.g., Pulido, Mandow, and Pérez-de-la-Cruz 2015, Ulloa et al. 2020). Khouadjia et al. (2013) evaluate a sub-optimal temporal MO planner, but their benchmarks all use makespan as one of two objectives and so are not usable in our setting. Therefore, we introduce the following MOP benchmark set.

5.1 Benchmarks

Among the IPC domains, one is actually an MO problem: in the SOKOBAN puzzle, the two traditional objectives are minimising the number of moves and the number of pushes. (The IPC domain considers only the latter.) We extend two other classical planning domains, DRIVERLOG and VISITALL, to MO problems, as follows: In VISITALL (a TSP-like problem on grids) we add multiple agents, with each agent's distance travelled as a separate cost function. Since agents are interchangeable, this creates a problem like the example from the introduction where the ideal point is always a zero vector. For DRIVERLOG (a logistics-like domain), we create three versions: DRIVERLOG-2 has the two objectives of minimising the total distance driven and the total distance walked (in the IPC version, the single objective is a linear combination of these two; note that there are a few other IPC domains whose objectives are also linear combinations of two or more terms, and which could be changed into MO problems in the same way); DRIVERLOG-4 has the two additional objectives of minimising the number of trucks and number of drivers used in the plan; finally, in DRIVERLOG-K each driver's distance walked and each truck's distance driven is a separate cost function.

Our second type of benchmark are all-outcomes determinisations of probabilistic planning problems, with unit cost and the negative logarithm of the outcome probability[1] as the two objectives. Thus, the two costs of a plan π are the length of π and a proxy for the probability of π failing to be executed in the probabilistic domain. However, π represents only one specific trajectory, so this is the probability of that trajectory failing to eventuate; it is possible that the plan succeeds also when some action(s) have a different outcome.

We apply this transformation to the BLOCKS WORLD, EXPLODING BW and T-TIREWORLD domains from IPPC'08 (Bryce and Buffet 2008). In T-TIREWORLD, we add an action `move-and-change-tire` combining the actions `move` and `change-tire`, if needed, in a row. This action is deterministic, i.e., has only one possible outcome, and its cost is $[1.5, 0]$, where 1.5 is the expectation of applying `change-tire` 50% of the time. Without this action, the PCS of each T-TIREWORLD task has only one plan, while in our modified version, it grows with task size.

For SOKOBAN, we use the "easy" and "medium" instances from Haslum et al. (2007); for EXPLODING BW, we use tasks from the IPPC, plus additional randomly generated tasks; for all other domains we generate tasks randomly. We exclude from the final benchmark set any task deemed too easy (solved by blind search in 10 seconds or less). Figure 2 shows distribution of the PCS size of tasks solved by at least one configuration, after filtering of trivial tasks.

[1]We approximate $-\log p_i$ by $\lfloor -1000 \times \log p_i \rfloor \in \mathbb{N}$.

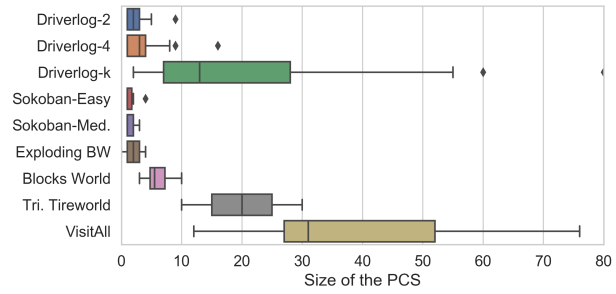

Figure 2: Boxplot of PCS size across the solved tasks in each domain. Omitted for clarity: DRIVERLOG-K has 5 other outliers between 86 and 154.

5.2 Experiments

We implemented a multi-objective planner in C++ based on the NAMOA* algorithm. Following insights from Ulloa et al. (2020) we modified the original algorithm to achieve better runtime performance; for details on the algorithm and the modifications we refer the reader to the corresponding PDF that is part of the planner and benchmark repository (Geißer et al. 2022). We used a timeout of 1800 seconds and a memory limit of 4GB; each planner run uses a single CPU core. Experiments were run on a cluster with Intel Xeon Platinum 8274 (Cascade Lake) 3.2 GHz CPUs. We used the python Lab package (Seipp et al. 2017); the implementation of operator counting constraints and PDB heuristics is based on the implementation in Fast Downward (Helmert 2006); the LP/IP solver used is CPLEX version 20.1.

We compare the multi-objective version of different heuristics to their ideal point counterparts. We also examine the effect of our two different MO maximisation operators in the MO heuristics that use them. We consider heuristic informativeness, measured by the number of node expansions, and heuristic effectiveness, measured by total runtime, required by our improved NAMOA* to compute the PCS.

Abstractions We consider the canonical PDB heuristic (H^C, cf. Section 3.3) using systematic pattern collections C that enumerate subsets of state variables, excluding redundant sets in the same way as Pommerening, Röger, and Helmert (2013). The MO heuristic computes PCSs for each PDB, then computes H^C. The ideal point heuristic uses the single-objective h^C on the same pattern collection.[2] We evaluated PDBs of size two and three; size three PDBs performed better in DRIVERLOG domains, but otherwise worse. Figures 3a and 3b compare expansions and runtime, respectively, of H^C with two variables per PDB using the two momax operators. Recall that comax is consistent while admax is not: the effect of inconsistency is visible

[2]We also tested a hybrid approach, which compute the ideal point heuristic for each abstraction and then uses H^C. This hybrid was inferior to the pure MO approach.

momax operator	blind	Critical Path				Abstractions					Operator Counting							
		$H^{h^{\max}}_{\text{ideal}}$	$H^{\max}_{\text{mo}}$ adm.	$H^{\max}_{\text{mo}}$ com.	H^{2}_{mo} com.	$H^{h^C_2}_{\text{ideal}}$	H^{C}_{2} adm.	H^{C}_{2} com.	$H^{h^C_3}_{\text{ideal}}$	H^{C}_{3} com.	$H^{h^+}_{\text{ideal}}$	$H^{h^+}_{\text{iter}}$	$H^{h^{\text{seq}}}_{\text{ideal}}$	$H^{h^{\text{seq}}}_{\text{iter}}$	$H^{h^+_{\text{lp}}}_{\text{ideal}}$	$H^{h^+_{\text{lp}}}_{\text{iter}}$	$H^{h^+_{\text{rel}}}_{\text{ideal}}$	$H^{h^{\text{seq}}_{\text{lp}}}_{\text{ideal}}$
DRIVERLOG-2 (52)	22	44	26	42	6	22	40	40	22	**49**	1	1	10	–	1	1	8	37
DRIVERLOG-4 (68)	18	**46**	12	37	2	18	29	31	18	43	–	–	4	–	1	1	1	17
DRIVERLOG-K (111)	22	46	14	39	1	22	43	43	22	**66**	–	–	37	2	10	1	25	43
SOKOBAN-E (26)	25	**26**	**26**	**26**	–	**26**	**26**	**26**	**26**	25	–	–	2	–	–	–	–	10
SOKOBAN-M (28)	–	**18**	11	13	–	12	**18**	**18**	9	10	–	–	–	–	–	–	–	2
EXPLODING BW (107)	13	46	33	46	24	26	**79**	**79**	26	70	39	34	14	–	38	12	31	25
BLOCKS WORLD (20)	6	10	5	5	–	6	14	**20**	6	18	10	3	19	13	6	1	14	**20**
T-TIREWORLD (11)	1	2	2	2	1	1	**11**	**11**	1	**11**	–	–	1	–	–	–	–	1
VISITALL (13)	3	4	**9**	6	6	3	7	7	3	6	–	–	–	–	–	–	–	3
Sum (436)	110	242	138	216	40	136	267	275	133	**298**	50	38	87	15	56	16	79	158

Table 1: Number of tasks solved by NAMOA* with different heuristics.

in BLOCKS WORLD, where using admax leads to significantly more node expansions. On the other hand, comax is more expensive to compute: this is seen in T-TIREWORLD and VISITALL, where using admax is significantly faster. Both these domains have large PCSs and also large sets of heuristic values. Figures 3c and 3d compare H^C and h^C, again for PDBs with two variables per pattern. Clearly, considering the MO aspect in the abstractions pays off. H^C_2 expands sometimes orders of magnitude fewer nodes, which often results in better runtime. The overhead of computing the MO heuristic is noticeable though: on tasks for which the number of expansions is similar, the ideal point heuristic often achieves better runtime (e.g., the SOKOBAN domain).

Critical Path Heuristics Figures 3e and 3f compare the two momax operators in the multi-objective formulation of $h^{\max}$ ($H^{\max}_{\text{mo}}$, cf. Section 3.4). Here, the negative effect of the inconsistency of admax is much greater, and using comax leads to significantly fewer expansions and a faster runtime in nearly all cases. Figures 3g and 3h compare $H^{\max}_{\text{mo}}$ to the ideal point variant. Again, the MO heuristic expands significantly fewer nodes, but overhead of its computation is greater than for PDBs, and for many tasks total runtime is higher than the ideal point heuristic. The exception is VISITALL, where the ideal point heuristic offers no information.

Operator Counting Heuristics We considered two types of operator counting constraints: the h^+ IP formulation by Imai and Fukunaga (2015), and net change constraints (van den Briel et al. 2007; Pommerening et al. 2014). To ensure admissibility, counting variables are integral (cf. Prop. 12). We therefore compare the iterative method of computing operator counting heuristics ($H^{h^+}_{\text{iter}}$ and $H^{h^{\text{seq}}}_{\text{iter}}$, respectively) with the corresponding ideal point heuristics which also constrain counting variables to be integral ($H^{h^+}_{\text{ideal}}$ and $H^{h^{\text{seq}}}_{\text{ideal}}$). Results are shown in Figures 3i, 3j, 3k and 3l. The integrality restriction makes these heuristics expensive, and they solve only few tasks. The iterative method has a significant overhead compared to the ideal point, as it has to solve many more IPs per heuristic evaluation.[3] The iterative method results in a more informed heuristic, requiring orders of magnitude fewer expansions, in the VISITALL domain, but both it and the ideal point heuristic with integral counting variables still solve only small tasks (those solved by blind search in less than 10 seconds).

Coverage The total number of tasks solved by each planner configuration is shown in Table 1. It also includes: blind search; H^2_{mo} (cf. Section 3.4; coverage with admax is omitted, as only 8 tasks were solved); PDBs of size 3 (H^C_3 and $H^{h^C_3}_{\text{ideal}}$; coverage with admax was generally worse, except in VISITALL where 8 tasks were solved); and three LP-relaxations of operator counting heuristics. The LP-relaxation of the ideal point heuristic using net change constraints ($H^{h^{\text{seq}}_{\text{lp}}}_{\text{ideal}}$) removes all integrality constraints. In the LP-relaxations using the h^+ constraints ($H^{h^+_{\text{lp}}}_{\text{ideal}}$ and $H^{h^+_{\text{lp}}}_{\text{iter}}$) counting variables are still integral, since this is necessary for the iterative method to be admissible, but other variables in the h^+ formulation are relaxed. Coverage generally correlates with the previous insights regarding expansions and runtime, and best performing are the MO variants of the canonical PDB heuristics.

Size of the Heuristic Set We also inspected the size of the heuristic set returned by each MO heuristic on the tasks they solve. The MO version of the IP-based heuristics return a small number of cost vectors: median of 1 and maximum of 2 vectors. In contrast, H^2_{mo} returned the largest number of cost vectors: the median, 75% quartile and maximum are 6, 7 and 132 (in an instance of DRIVERLOG-4) when using admax and 5, 14 and 1,666 (in an instance of DRIVERLOG-K) when using comax. The MO PDB heuristics seem to achieve a good trade-off between computational cost and number of returned vectors: H^C_2 median was 1, maximum was 18 and it returned well above 5 vectors for several tasks, using both admax and comax; and H^C_3 median, 75%-quartile and maximum were 2, 3, and 23, when

[3] We also devised a weighted sum-based MOLP solver and an efficient discretisation algorithm (cf. Section 3.5) for bi-objective problems. We do not include results since this approach currently is applicable only to the subset of bi-objective tasks. How to compute the discretisation of the MOLP Pareto front efficiently in general (for $k > 2$) is an open problem, which we leave for future work.

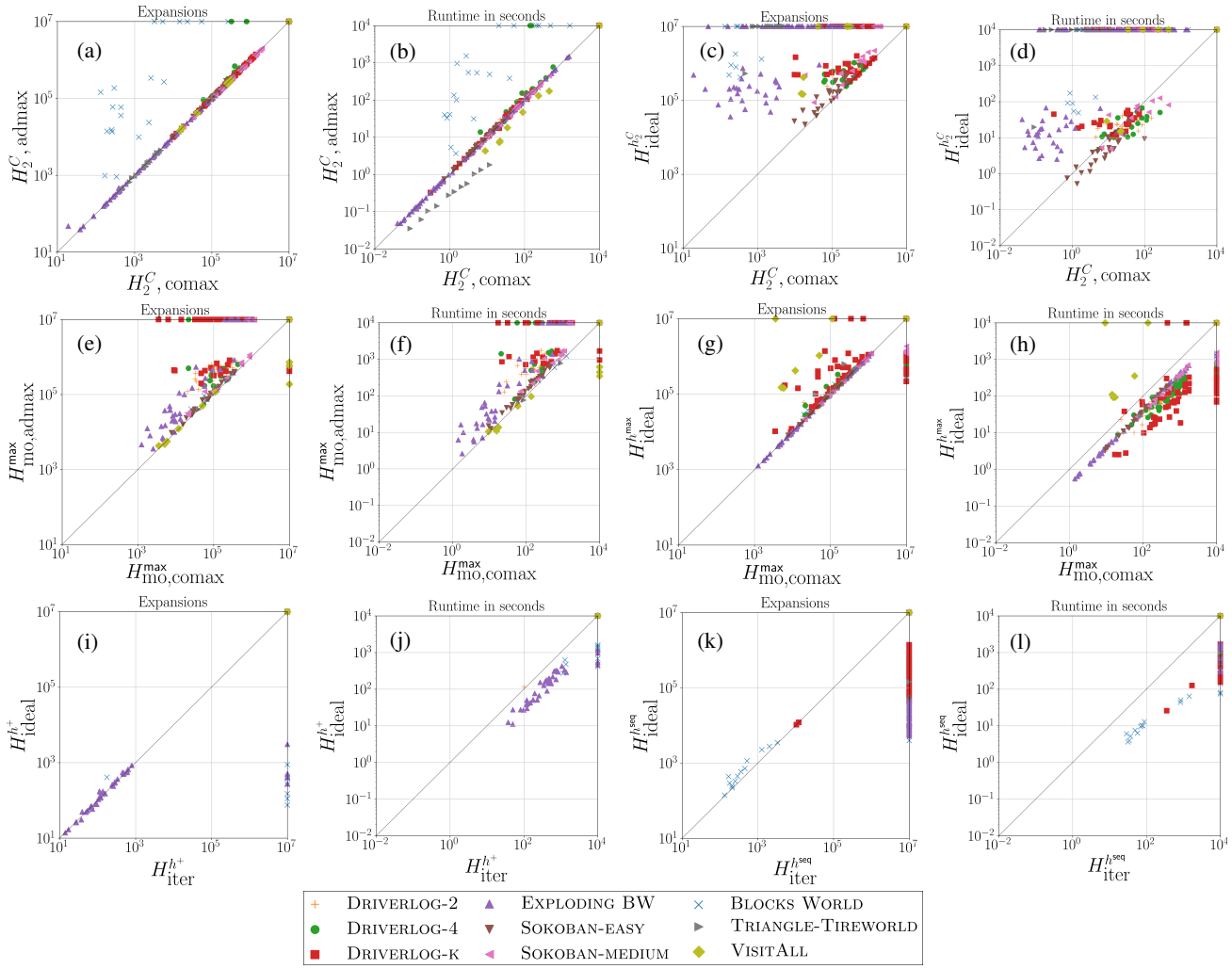

Figure 3: Comparison of number of NAMOA* node expansions and runtime between the different heuristics. Data points on axis limits correspond to runs where runtime or memory was exceeded.

using admax and 1, 2, and 20 when using comax.

6 Conclusion

We have shown that extending admissible heuristics from classical planning into genuine MO heuristics that capture objective trade-offs can, with the right heuristic for the right task, greatly improve the performance of MO planning via search, but also raises several questions. The MO heuristics are often more informed than their respective ideal point combination, but this is not always true and our understanding of the properties of the task that determine when it happens is incomplete. The impact of heuristic inconsistency on NAMOA* appears greater than what is usually seen in single-objective A*. The cause of inconsistency in the heuristics we considered is one of our MO maximisation operators, admax. Since we have shown that the maximum of MO heuristic sets is not unique, we may yet ask whether there are still better momax operators to be discovered.

In this paper, we have evaluated only basic representatives of some of the different families of heuristics, for example the canonical PDB heuristic. More advanced classical heuristic constructions, e.g., recently proposed cost partitioning schemes for abstractions (Seipp, Keller, and Helmert 2017), are likely to yield stronger MO heuristics as well, but because those constructions exploit information about action costs, they will require greater changes to adapt.

Some of the MO heuristics are far too expensive per node to evaluate. This is true of the iterative method of computing operator counting heuristics in particular. However, the operator counting framework holds great potential for integrating other problem aspects into heuristics, notably probabilities (Trevizan, Thiébaux, and Haslum 2017) for solving multi-objective stochastic shortest path problems (MO-SSPs) (Roijers et al. 2013) with generalisations of MO heuristic search. Hence, devising general methods of solving the operator counting problem that better balance time and pruning power is an important open problem, which we have only begun to investigate.

Acknowledgements

This work was supported by ARC project DP180103446, *On-line planning for constrained autonomous agents in an uncertain world* and by computational resources provided by the Australian Government through the National Computational Infrastructure (NCI) under the ANU Startup Scheme. We thank Florian Pommerening for access to his implementation of LP-based h^+.

References

Ahmadi, S.; Tack, G.; Harabor, D.; and Kilby, P. 2021. Bi-Objective Search with Bi-Directional A^*. In *Proc. ESA 2021*, 3:1–3:15.

Bonet, B.; and Geffner, H. 1999. Planning as Heuristic Search: New Results. In *Proc. ECP 1999*, 360–372.

Bryce, D.; and Buffet, O. 2008. 6th Int. Planning Competition: Uncertainty Track. In *3rd Int. Probabilistic Planning Competition (IPPC-ICAPS'08)*.

Edelkamp, S. 2001. Planning with Pattern Databases. In *Proc. ECP 2001*, 84–90.

Ehrgott, M. 2005. *Multicriteria Optimization*. Springer. ISBN 3-540-21398-8.

Fikes, R. E.; and Nilsson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *AIJ*, 2: 189–208.

Geißer, F.; Haslum, P.; Thiébaux, S.; and Trevizan, F. 2022. Code and benchmarks for the ICAPS 2022 paper "Admissible Heuristics for Multi-Objective Planning". https://doi.org/10.5281/zenodo.6383217.

Grastien, A.; Benn, C.; and Thiébaux, S. 2021. Computing Plans that Signal Normative Compliance. In *Proc. AIES 2021*, 509–518.

Haslum, P.; Bonet, B.; and Geffner, H. 2005. New Admissible Heuristics for Domain-Independent Planning. In *Proc. AAAI 2005*, 1163–1168.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-Independent Construction of Pattern Database Heuristics for Cost-Optimal Planning. In *Proc. AAAI 2007*, 1007–1012.

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

Helmert, M. 2006. The Fast Downward Planning System. *JAIR*, 26: 191–246.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces. *JACM*, 61(3): 16:1–63.

Imai, T.; and Fukunaga, A. 2015. On a Practical, Integer-Linear Programming Model for Delete-Free Tasks and its Use as a Heuristic for Cost-Optimal Planning. *JAIR*, 54: 631–677.

Karpas, E.; and Domshlak, C. 2009. Cost-Optimal Planning with Landmarks. In *Proc. IJCAI 2009*, 1728–1733.

Katz, M.; and Domshlak, C. 2008. Optimal Additive Composition of Abstraction-based Admissible Heuristics. In *Proc. ICAPS 2008*, 174–181.

Khouadjia, M.; Schoenauer, M.; Vidal, V.; Dréo, J.; and Savéant, P. 2013. Pareto-Based Multiobjective AI Planning. In *Proc. IJCAI 2013*, 2321–2327.

Machuca, E.; Mandow, L.; Pérez-de-la-Cruz, J.; and Ruiz-Sepúlveda, A. 2009. An Empirical Comparison of Some Multiobjective Graph Search Algorithms. In *Proc. KI 2009*, 238–245.

Machuca, E.; Mandow, L.; Pérez-de-la-Cruz, J.; and Ruiz-Sepúlveda, A. 2012. A comparison of heuristic best-first algorithms for bicriterion shortest path problems. *European Journal of Operational Research*, 217(1): 44–53.

Mandow, L.; and Pérez-de-la-Cruz, J. 2010. Multiobjective A* search with consistent heuristics. *J. ACM*, 57(5): 27:1–27:25.

Nebel, B. 2000. On the Compilability and Expressive Power of Propositional Planning Formalisms. *Journal of AI Research*, 12: 271–315.

Pommerening, F.; Röger, G.; and Helmert, M. 2013. Getting the Most Out of Pattern Databases for Classical Planning. In *Proc. IJCAI 2013*, 2357–2364.

Pommerening, F.; Röger, G.; Helmert, M.; and Bonet, B. 2014. LP-based Heuristics for Cost-optimal Planning. In *Proc. ICAPS 2014*, 226–234.

Pulido, F. J.; Mandow, L.; and Pérez-de-la-Cruz, J. 2015. Dimensionality reduction in multiobjective shortest path search. *Computers & Operations Research*, 64: 60–70.

Roijers, D. M.; Vamplew, P.; Whiteson, S.; and Dazeley, R. 2013. A Survey of Multi-Objective Sequential Decision-Making. *JAIR*, 48: 67–113.

Scala, E.; Haslum, P.; and Thiébaux, S. 2016. Heuristics for Numeric Planning via Subgoaling. In *Proc. IJCAI 2016*, 3228–3234.

Seipp, J.; Keller, T.; and Helmert, M. 2017. A Comparison of Cost Partitioning Algorithms for Optimal Classical Planning. In *Proc. ICAPS 2017*, 259–268.

Seipp, J.; Pommerening, F.; Sievers, S.; and Helmert, M. 2017. Downward Lab. https://doi.org/10.5281/zenodo.790461.

Sievers, S.; Ortlieb, M.; and Helmert, M. 2012. Efficient Implementation of Pattern Database Heuristics for Classical Planning. In *Proc. SoCS 2012*, 105–111.

Stewart, B. S.; and White, C. C., III. 1991. Multiobjective A*. *JACM*, 38(4): 775–814.

Trevizan, F. W.; Thiébaux, S.; and Haslum, P. 2017. Occupation Measure Heuristics for Probabilistic Planning. In *Proc. ICAPS 2017*, 306–315.

Ulloa, C. H.; Yeoh, W.; Baier, J. A.; Zhang, H.; Suazo, L.; and Koenig, S. 2020. A Simple and Fast Bi-Objective Search Algorithm. In *Proc. ICAPS 2020*, 143–151.

van den Briel, M.; Benton, J.; Kambhampati, S.; and Vossen, T. 2007. An LP-Based Heuristic for Optimal Planning. In *Proc. CP 2007*, 651–665.