Cost Partitioning for Multiple Sequence Alignment

Mika Skjelnes, Daniel Gnad, Jendrik Seipp

Linköping University, Sweden
firstname.lastname@liu.se

Abstract

Multiple Sequence Alignment (MSA) is a fundamental problem in computational biology that is used to understand the evolutionary history of protein, DNA, or RNA sequences. An optimal alignment for two sequences can efficiently be found using dynamic programming, but computing optimal alignments for more sequences continues to be a hard problem. A common method to solve MSA problems is A* search with admissible heuristics, computed from subsets of the input sequences. In this paper, we consider MSA from the perspective of cost partitioning and relate the existing heuristics for MSA to uniform cost partitioning and post-hoc optimization, two well-known techniques from the automated planning literature. We show that the MSA heuristics are bounded by uniform cost partitioning and that post-hoc optimization yields strictly dominating heuristics. For a common benchmark set of protein sequences and a set of DNA sequences, we show that the theoretical dominance relations between the heuristics carry over to practical instances.

Introduction

The Multiple Sequence Alignment (MSA) problem is the task of optimally aligning a set of character sequences based on a pairwise character alignment cost (Kobayashi and Imai 1998). An *alignment* is a modification of the sequences where gaps are inserted into a sequence to offset its tail, so that all sequences have the same length. The cost of an alignment is computed by summing up the individual character alignment costs across the characters at the same index over all sequence pairs. See Figure 1 for an example. MSA is a central problem in molecular biology, and is relevant, among others, for research of relationships in protein structure and evolutionary studies (Carrillo and Lipman 1988).

MSA can be solved optimally with dynamic programming (Needleman and Wunsch 1970), but this method becomes impractical as the number of sequences grows. Hence, a typical approach to solve MSA is based on solving selected sub-problems with dynamic programming, and combining the solutions of these sub-problems to form an *admissible* heuristic (Pearl 1984). This heuristic is then used to guide an A^* search (Hart, Nilsson, and Raphael 1968) for solving the full problem. Since their introduction in 1998, the strongest admissible heuristics for MSA are $h^{\text{all},k}$ and $h^{\text{one},k}$ (Kobayashi and Imai 1998). They both dominate the h^{pair} heuristic (Ikeda and Imai 1994) which sums over all pairwise alignment costs.

S			
s_1:	T	T	A
s_2:	G	C	
s_3:	A	C	

A				
A_1:	T	T	–	A
A_2:	–	G	–	C
A_3:	A	–	C	–

Figure 1: Three sequences S and an alignment A thereof.

We build upon previous research (Riesterer 2018) on relating the $h^{\text{all},k}$ and $h^{\text{one},k}$ heuristics to the concept of *cost partitioning*. Cost partitioning (CP) is the state-of-the-art approach in optimal classical planning to combine heuristics admissibly (Yang et al. 2008; Katz and Domshlak 2010; Pommerening et al. 2015; Franco et al. 2017; Seipp, Keller, and Helmert 2020). By distributing the cost of each action across the heuristics, we can sum the heuristic estimates while guaranteeing admissibility. Cost partitioning has been studied extensively, and various methods have been developed for partitioning action costs (Haslum, Bonet, and Geffner 2005; Haslum et al. 2007; Katz and Domshlak 2008; Karpas and Domshlak 2009; Katz and Domshlak 2010; Pommerening, Röger, and Helmert 2013; Seipp and Helmert 2014; Seipp, Keller, and Helmert 2017, 2021). For some of these methods, dominance relationships have been established, i.e., theoretical guarantees that one method will always yield an overall heuristic value at least as high as the one of the other, when applied to the same set of heuristics (Seipp, Keller, and Helmert 2017).

In this work, we view MSA problems as shortest-path searches in transition systems. We establish a formal connection between $h^{\text{all},k}$ and $h^{\text{one},k}$ to cost partitioning over a collection of abstraction heuristics (Katz and Domshlak 2010; Pommerening, Röger, and Helmert 2013). Concretely, we show that, when using the same collection of heuristics, $h^{\text{all},k}$ and $h^{\text{one},k}$ are dominated by *uniform cost partitioning* (UCP) (Katz and Domshlak 2010), which gives an equal share of the cost of each action to every heuristic it contributes to. For *post-hoc optimization* (PhO) (Pommerening, Röger, and Helmert 2013), which maximizes the sum over the heuristics by computing the weight of each heuristic using linear programming, we get the stronger result that it even strictly dominates $h^{\text{all},k}$. This still ignores the fact that

both cost-partitioning approaches are capable of handling arbitrary collections of heuristics, in contrast to the established MSA heuristics $h^{\text{all},k}$ and $h^{\text{one},k}$, which work on a fixed collection.

We evaluate the new post-hoc optimization heuristic on a core bio-informatics benchmark set of protein sequences called *BAliBase* (Thompson, Plewniak, and Poch 1999), as well as a set of randomly generated DNA sequences. Our results confirm empirically that cost partitioning is indeed promising for MSA, with post-hoc optimization yielding higher heuristic values than the previous state-of-the-art heuristics.

Multiple Sequence Alignment

Given a set of sequences $S = \{s_1, \ldots, s_n\}$ over some alphabet Σ, an *alignment* of S is a matrix $A^{n \times m}$ over alphabet $\Sigma' = \Sigma \cup \{-\}$, such that the following conditions hold: (1) $m \geq \max_{s_i \in S} |s_i|$, (2) no column j consists of only the *gap* character "–", and (3) removing all gap occurrences from any row i in A yields the original sequence s_i. The *cost* of an alignment is the summed pairwise substitution cost $sub : \Sigma' \times \Sigma' \to \mathbb{N}_0^+$ of each two characters per column in the alignment across all columns. The concrete substitution cost function is problem-specific, and can be based on biological knowledge or empirical data.

Example 1. *Given sequences $S = \{s_1, s_2, s_3\}$ where $s_1 =$ `TTA`, $s_2 =$ `GC`, $s_3 =$ `AC`, we can construct an alignment A with rows $A_1 =$ `TT--A`, $A_2 =$ `--G--C`, $A_3 =$ `A--C--`, visualized in Figure 1. We will use S and A as our running example.*

Formally, the alignment costs of two rows of an alignment, respectively the full alignment, are defined as follows.

Definition 1 (Pair Score)**.** *Let $s_i, s_j \in S$, with $i \neq j$, be two sequences over alphabet Σ, and A an alignment of S. Then the pair score of s_i and s_j in A is*

$$C(A, i, j) = \sum_{k=1}^{m} sub(a_{ik}, a_{jk}).$$

Definition 2 (Alignment Score)**.** *The* alignment score*, or* sum of pairs score *of alignment A is*

$$C(A) = \sum_{1 \leq i < j \leq n} C(A, i, j).$$

We also define a score function for an alignment *column*.

Definition 3 (Column Score)**.** *The* column score *of a column $\mathcal{C}$ of alignment A is defined as*

$$C(\mathcal{C}) = \sum_{1 \leq i < j \leq n} sub(\mathcal{C}_i, \mathcal{C}_j).$$

The goal of MSA is to find an optimal alignment A, i.e., one where for all alignments $A' : C(A) \leq C(A')$.

Example 2. *Consider sequences S, and alignment A from the running example. The pair score of A_1 and A_2 is $C(A, 1, 2) = sub(T, -) + sub(T, G) + sub(-, -) + sub(A, C)$. The alignment score of A is $C(A, 1, 2) + C(A, 1, 3) + C(A, 2, 3)$.*

MSA as Shortest-Path Search

Previous work has already discussed that an MSA task can be seen as a shortest-path search problem in a weighted transition system (e.g., Kobayashi and Imai 1998). However, this connection has so far been left implicit in the literature. We make this perspective explicit by defining the transition system for an MSA task and showing how the cost of a path in this transition system corresponds to the cost of an alignment. Figure 2 visualizes the transition system for the running example.

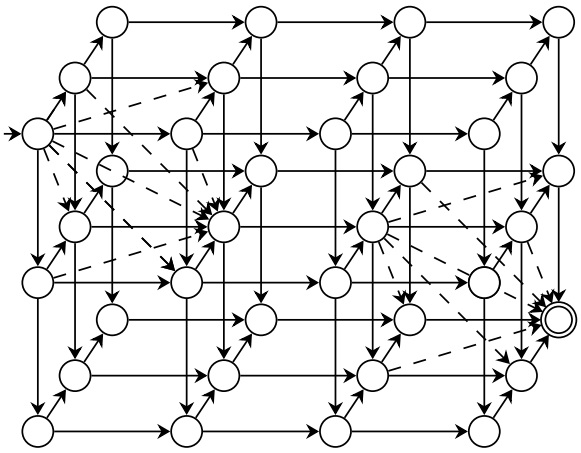

Figure 2: Transition system for the running example, where three sequences of length 3 (x axis), 2 (y axis) and 2 (z axis) need to be aligned. A shortest path from the initial state in the front top left to the goal state at the back lower right defines an MSA. To avoid clutter, we omit a significant number of transitions. For the top left and bottom right cube, we show all transitions. Dashed transitions make progress in more than one sequence at a time.

Definition 4 (Transition System)**.** *A* transition system $\mathcal{T}$ *is a directed labeled graph defined by finite set of states $S(\mathcal{T})$, a finite set of labels $L(\mathcal{T})$, a set $T(\mathcal{T})$ of labeled transitions $s \xrightarrow{\ell} s'$ with $s, s' \in S(\mathcal{T})$ and $\ell \in L(\mathcal{T})$, an initial state $s_0(\mathcal{T})$, and a set $S_*(\mathcal{T})$ of goal states.*

For an MSA task with n sequences $s_1, \ldots s_n$, we obtain the corresponding transition system $\mathcal{T}$ as follows. The states $S(\mathcal{T})$ are the vertices in an n-dimensional lattice graph $\mathcal{G}$ of size $|s_1 + 1| \times \cdots \times |s_n + 1|$. The initial state $s_0(\mathcal{T})$ is at the origin $\langle 0, \ldots, 0 \rangle$, and the single goal state s_* is at coordinate $\langle |s_1|, \ldots, |s_n| \rangle$.

There is a transition $s \xrightarrow{\ell} s'$ between state $s = \langle x_1, x_2, \ldots, x_n \rangle$ and state $s' = \langle x'_1, x'_2, \ldots, x'_n \rangle$ if, and only if, between the two states there is at most a single step of progress in every dimension, and at least one step of progress in one dimension. Formally, this is the case if $0 \leq x'_i - x_i \leq 1$ for all $i = 1 \ldots n$, and $\sum_{i=1}^{n} (x'_i - x_i) > 0$.

In transition systems for other search tasks, each transition is directly labeled with the cost of taking that transition, or with an action label that indirectly defines the cost. For

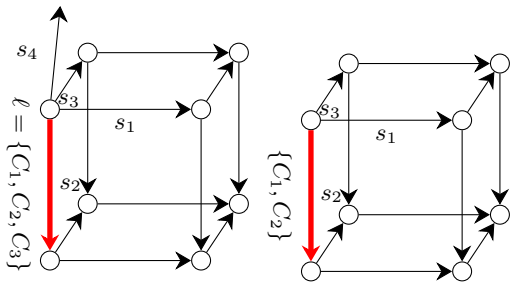

Figure 3: Cost component decomposition of label ℓ in the transition system of the full task (left), and a transition system $\mathcal{T}_{P_1}$ for pattern $P_1 = \{s_1, s_2, s_3\}$ (right).

MSA tasks, however, the definition of alignment costs requires us to associate each transition label with a *set* of *cost components*.

Definition 5 (Cost Components)**.** *Let s_i and s_j be two sequences in an MSA task. Then the* cost component $C^{i,j}_{\langle x,y\rangle \to \langle x',y'\rangle}$ *is the cost of the interaction between sequences s_i and s_j from positions $\langle x, y\rangle$ to $\langle x', y'\rangle$, where $0 \leq x \leq x' \leq |s_i|$ and $0 \leq y \leq y' \leq |s_j|$.*

Using this definition, we label each transition $t \in T(\mathcal{T})$ with the set of cost components relevant for t. This corresponds to associating each transition t with the set of substitution costs for the corresponding column of the alignment.

Definition 6 (Paths and Goal Paths)**.** *Let $\mathcal{T}$ be a transition system. A* path *from $s \in S(\mathcal{T})$ to $s' \in S(\mathcal{T})$ is a sequence of transitions from $T(\mathcal{T})$ of the form $\pi = \langle s^0 \xrightarrow{\ell_1} s^1, s^1 \xrightarrow{\ell_2} s^2, \ldots, s^{n-1} \xrightarrow{\ell_n} s^n\rangle$, where $s^0 = s$ and $s^n = s'$. A path is a* goal path *if $s^0 = s_0$ and s^n is a goal state of $\mathcal{T}$.*

Intuitively, each goal path in the transition system is an alignment. Each step in the goal path corresponds to one column in the final alignment matrix. As such, each step makes progress on a subset $S' \subseteq S$ by aligning the characters at their current positions. For the remaining sequences, the step inserts a gap in the corresponding column of the alignment. The cost of each transition is the column score of the column formed by the transition, or equivalently, the sum of the cost components of the transition.

Example 3. *Consider sequences $S = \{s_1, \ldots, s_4\}$ (not the running example), transition system $\mathcal{T}$ induced by the MSA task of aligning S, and transition $t = \langle 0,0,0,0\rangle \to \langle 0,1,0,0\rangle \in T(\mathcal{T})$. Then t is labeled with the set of cost components $\ell = \{C_1, C_2, C_3\}$, where $C_1 = C^{1,2}_{\langle 0,0\rangle \to \langle 0,0\rangle}$, $C_2 = C^{2,3}_{\langle 0,0\rangle \to \langle 1,0\rangle}$ and $C_3 = C^{2,4}_{\langle 0,0\rangle \to \langle 1,0\rangle}$. Figure 3 visualizes the cost components of t in $\mathcal{T}$, and in the transition system $\mathcal{T}_{\{s_1,s_2,s_3\}}$ induced by the MSA task of aligning $\{s_1, s_2, s_3\}$. Figure 4 visualizes the cost decomposition of ℓ in $\mathcal{T}_{\{s_1,s_2\}}$, $\mathcal{T}_{\{s_2,s_3\}}$, and $\mathcal{T}_{\{s_2,s_4\}}$.*

Definition 7 (Cost Functions)**.** *A* cost function *for transition system $\mathcal{T}$ is a function cost $: L(\mathcal{T}) \to \mathbb{R}_0^+$. In the case of MSA transition systems, where each transition is labeled with a set of cost components, the cost of a transition t is the sum of its cost components. The cost of a path π in $\mathcal{T}$ is the sum of the cost of its transitions.*

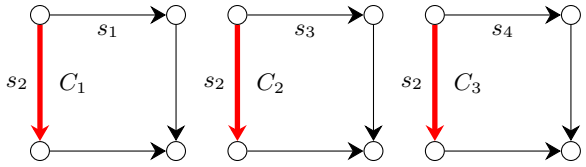

Figure 4: Transition systems $\mathcal{T}_{P_2}, \mathcal{T}_{P_3}, \mathcal{T}_{P_4}$ for patterns $P_3 = \{s_1, s_2\}, P_3 = \{s_2, s_3\}, P_4 = \{s_2, s_4\}$, respectively. The label for the (red) transition is decomposed to a single cost component in each transition system here.

A *cheapest goal path* π in $\mathcal{T}$ defines an MSA solution and we use $h^*_{\mathcal{T}}(s_0(\mathcal{T}))$ to refer to its cost. Each transition $t_i \in \pi$ corresponds to column $\mathcal{C}_i$ in the resulting alignment A.

Example 4. *Consider sequences S and alignment A from the running example. Aligning the first character of sequence s_1 with the first character of sequence s_3, and inserting a gap for sequence s_2, results in the first column of alignment A. This corresponds to taking the transition $\langle 0,0,0\rangle \to \langle 1,0,1\rangle$ in the induced transition system. The cost of this transition is the column score of col, which is the first column $\langle$T-A$\rangle$ of A: $C(col) = sub(T,-) + sub(T,A) + sub(-,A)$.*

Pattern Database Heuristics for MSA

Now, we can solve MSA by finding a cheapest goal path π in the induced transition system $\mathcal{T}$. In principle, one could use uninformed search algorithms like *uniform cost search* to find cheapest paths. However, the enormous size of $\mathcal{T}$, which has $\prod_{s \in S}(|s|+1)$ states for sequences S, makes such blind approaches infeasible for realistic MSA tasks. Instead, we turn to heuristic search and use A* to explore the state space guided by an admissible heuristic. This approach guarantees that the found paths are optimal.

The most prominent way of obtaining an admissible heuristic for MSA transition system $\mathcal{T}$ is to select a subset of sequences $P \subseteq S$ such that the transition system $\mathcal{T}'$ for aligning P is small enough to be explored exhaustively (Kobayashi and Imai 1998). This approach is closely related to the concept of *abstraction* and *projection* in heuristic search (Felner, Korf, and Hanan 2004). More precisely, each subset P can be seen as a *pattern* and aligning the sequences in the pattern is similar to finding a shortest path in the MSA transition system projected to the dimensions present in P. Computing the costs of optimal goal paths in such a projection gives rise to a *pattern database heuristic*, which are widely used in heuristic search (Culberson and Schaeffer 1998; Edelkamp 2001). However, due to the way costs are defined for MSA, we cannot define MSA projections by homomorphic abstraction as is common for heuristic search tasks (Seipp and Helmert 2018). Instead, we define the projection to P as the transition system that we obtain for the sequences P. This is identical to removing all cost com-

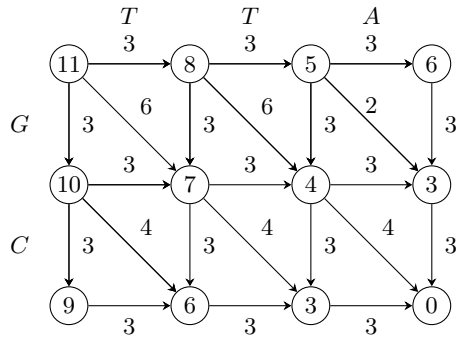

Figure 5: Abstract transition system, of the full transition system in Figure 2, induced by the task of aligning sequences s_1 and s_2 in the running example. For each abstract state, we show its optimal goal distance.

ponents that involve sequences not in P when we project the full MSA lattice to the dimensions in P.

Definition 8 (Pattern Database Heuristics for MSA)**.** *Let $P = \{s_1, \ldots, s_k\} \subseteq S$ be a subset of sequences in an MSA task. Then we call P a* pattern *and let the* projection $\mathcal{T}_P$ *refer to the MSA transition system for aligning P. For a state s, we define $h^P(s)$ as the cost of a cheapest goal path in $\mathcal{T}_P$ starting in state $s_{|P}$, where $s_{|P}$ is obtained by projecting s to the sequences in P. Where convenient, we write $h^P(s)$ as $h^{s_1,\ldots,s_k}(s)$.*

Example 5. *In our running example, consider the subproblem of aligning $P = \{s_1, s_2\}$. Figure 5 shows the induced transition system $\mathcal{T}_P$ for this abstraction and the goal distance of each abstract state in $\mathcal{T}_P$. For example, for the initial state s_0 of the full task, the heuristic estimate we obtain from $\mathcal{T}_P$ is $h^{s_1,s_2}(s_0) = h^*_{\mathcal{T}'}(s_{0|P}) = 11$.*

Instead of using a single pattern database heuristic, we can combine multiple heuristics to obtain a more informed estimate. This is also done by the two state-of-the-art MSA heuristics $h^{\text{all},k}$ and $h^{\text{one},k}$. They both select sequence subsets and compute heuristics over the induced abstractions (Kobayashi and Imai 1998). As such, they can be seen as examples for pattern database heuristics. The $h^{\text{all},k}$ heuristic computes the sum of abstract goal distances obtained from all k-fold patterns. To ensure admissibility, the sum is divided by the number of k-fold patterns in which two sequences s_i, s_j appear together, which amounts to $\binom{n-2}{k-2}$ patterns.

Definition 9 ($h^{\text{all},k}$)**.** *Given an MSA task with n sequences $S = \{s_1, \ldots, s_n\}$ and a natural number k such that $2 \leq k \leq n$, the $h^{\text{all},k}$ heuristic is defined as*

$$h^{\text{all},k}(s) = \frac{1}{\binom{n-2}{k-2}} \sum_{1 \leq x_1 < \cdots < x_k \leq n} h^{s_{x_1},\ldots,s_{x_k}}(s).$$

Computing all k-folds can however be too computationally demanding. The $h^{\text{one},k}$ heuristic can be faster to compute than $h^{\text{all},k}$, as it only solves two larger subproblems $h^{s_{x_1},\ldots,s_{x_k}}$ and $h^{s_{x_{k+1}},\ldots,s_{x_n}}$, in addition to all pair patterns obtained by taking exactly one sequence from each of the two larger subproblems.

Definition 10 ($h^{\text{one},k}$)**.** *Given an MSA task with n sequences $S = \{s_1, \ldots, s_n\}$, a natural number k such that $2 \leq k \leq n$, a k-fold pattern $P_1 = \{s_{x_1}, \ldots s_{x_k}\}$ and an $(n-k)$-fold pattern $P_2 = \{s_{x_{k+1}}, \ldots s_{x_n}\}$ so that $P_1 \cup P_2 = S$ and $P_1 \cap P_2 = \emptyset$, the $h^{\text{one},k}$ heuristic is defined as*

$$h^{\text{one},k}(s) = h^{s_{x_1},\ldots,s_{x_k}} + h^{s_{x_{k+1}},\ldots,s_{x_n}} + \sum_{i=1}^{k} \sum_{j=k+1}^{n} h^{s_{x_i},s_{x_j}}.$$

Since $h^{\text{one},k}$ only considers collections of pattern database heuristics whose sum is admissible (Kobayashi and Imai 1998), scaling the estimate is unnecessary.

Cost Partitioning

Cost Partitioning is a prominent technique from the classical planning and heuristic search literature, which allows to sum heuristic estimates while preserving admissibility (Yang et al. 2008; Katz and Domshlak 2010). A *cost partitioning* distributes the cost of each action among the component heuristics, such that the sum of costs per action is not greater than the original cost.

Definition 11 (Cost Partitioning)**.** *Given a tuple of n heuristics $\mathcal{H} = \langle h_1, \ldots, h_n \rangle$ for transition system $\mathcal{T}$ with labels $L(\mathcal{T})$, and cost function c, the cost functions $\mathcal{C} = \langle c_1, \ldots, c_n \rangle$ form a* cost partitioning *for $\mathcal{H}$ if $\sum_{i=1}^{n} c_i(\ell) \leq c(\ell)$ for all $\ell \in L(\mathcal{T})$. The resulting cost-partitioned heuristic is $h^{\mathcal{C}}(s) = \sum_{i=1}^{n} h_i(c_i, s)$, where $h_i(c_i, s)$ is the heuristic value of h_i for s evaluated under cost function c_i.*

Computing an *optimal cost partitioning* (Katz and Domshlak 2010; Pommerening et al. 2021) over abstraction heuristics is possible in polynomial time, but still usually prohibitively expensive in practice (Seipp, Keller, and Helmert 2020). Instead, we turn to computationally less demanding algorithms for computing cost partitionings, which we introduce next. In the definitions below, we say that a label ℓ is *used* by an abstraction heuristic h_i if ℓ induces a state-changing transition in $\mathcal{T}_i$, the transition system underlying h_i.

Uniform Cost Partitioning A *uniform cost partitioning* (UCP) distributes the cost of each label ℓ evenly among all heuristics that use ℓ (Katz and Domshlak 2010). Formally, cost function c_i for heuristic h_i is defined as

$$c_i(\ell) = \begin{cases} \frac{c(\ell)}{|\{h \in \mathcal{H} | h \text{ uses } \ell\}|} & \text{if } h_i \text{ uses } \ell \\ 0 & \text{otherwise.} \end{cases}$$

We refer to the resulting uniform cost-partitioned heuristic as h_{UCP}.

Post-Hoc Optimization *Post-hoc optimization* (PhO) (Pommerening, Röger, and Helmert 2013; Seipp, Keller, and Helmert 2021) uses a *linear program* (LP) to compute a weight w_i for each heuristic $h_i \in \mathcal{H}$ such that the weighted

sum of heuristic estimates remains admissible:

$$h_{\text{PhO}}(s) = \text{maximize} \sum_{i=1}^{n} w_i \cdot h_i(s) \text{ s.t.}$$
$$\sum_{h_i \in \mathcal{H}: h_i \text{ uses } \ell} w_i \leq 1 \text{ for all } \ell \in L(\mathcal{T})$$
$$w_i \geq 0 \text{ for all } h_i \in \mathcal{H}.$$

The resulting post-hoc optimization cost partitioning is the tuple $\mathcal{C} = \langle w_1 \cdot c_1, \ldots, w_n \cdot c_n \rangle$, where $c_i(\ell) = c(\ell)$ if h_i uses ℓ and $c_i(\ell) = 0$ otherwise.

Definition 12 (Dominance)**.** *Heuristic h* dominates *heuristic h' if $h(s) \geq h'(s)$ for all states $s \in S(\mathcal{T})$. The dominance is* strict *if there is a state s such that $h(s) > h'(s)$.*

There is no dominance relation between h_{UCP} and h_{PhO}, i.e., there are example transition systems $\mathcal{T}$ and states $s, s' \in S(\mathcal{T})$, where $h_{\text{UCP}}(s) > h_{\text{PhO}}(s)$ and $h_{\text{PhO}}(s) > h_{\text{UCP}}(s)$. Below, we will establish equality and dominance relations between the MSA heuristics and cost-partitioned heuristics.

Additive MSA Abstractions

The notion of cost components allows us to identify how different projections share costs on their labels. We will next show that this is essential to reason about admissibility of the sum over pattern database heuristics. When summing up costs across several projections we can connect the occurrence of cost components directly to admissibility: if a cost component C appears in at most one projection $P \in \mathcal{P}$, then the respective heuristics are additive, i.e. their sum is admissible.

Definition 13 (Additive Pattern Collection)**.** *A collection of pattern database heuristics $\mathcal{P}$ is* additive *iff for all states $s \in S(\mathcal{T})$: $h^{\mathcal{P}}(s) := \sum_{P \in \mathcal{P}} h^P(s) \leq h^*_{\mathcal{T}}(s)$.*

First, we prove that every cost component can appear at most once along every solution.

Proposition 1. *Let $\mathcal{T}$ be the transition system induced by a MSA task and π a goal path of $\mathcal{T}$. Then every cost component C is part of at most one transition label in π.*

Proof. Without loss of generality, let $C = C^{i,j}_{\langle x,y \rangle \to \langle x',y' \rangle}$ be contained in a label for transition t in π. By definition, every cost component labels a transition t' in $\mathcal{T}$ that starts at coordinate $\langle x, y \rangle$ and ends in $\langle x', y' \rangle$, where $x' > x \vee y' > y$, affecting dimensions i and j. Since every $\mathcal{T}$ induced by a MSA task is acyclic, the progress made from x to x' (resp. y to y') cannot be undone, so C cannot appear again on π. □

With this, across the solutions of a set of transition systems $\mathcal{T}_P$, we know that if a cost component C appears more than once, then its cost is over-counted when combining the solutions, i.e. when summing up the heuristics. Thus, if every cost component appears only on labels of a single pattern database heuristic from a collection $\mathcal{P}$, then the collection is additive.

Proposition 2. *Let $\mathcal{P}$ be a pattern collection. If for every cost component C there exists at most one pattern $P \in \mathcal{P}$ such that C appears in the induced transition systems $\mathcal{T}_P$, then $\mathcal{P}$ is additive.*

Proof. Let π_P denote a solution for $\mathcal{T}_P$ of a pattern $P \in \mathcal{P}$. Then, because every cost component C appears in at most one projection P, the set of solutions $\{\pi_P \mid P \in \mathcal{P}'\}$ of any subset $\mathcal{P}' \subseteq \mathcal{P}$ consider C no more than once. Hence, the heuristic values h^P for all $P \in \mathcal{P}'$ can be summed up, leading to an admissible heuristic. □

A pattern collection that uses patterns with cost components in common is not necessarily additive by itself, but a heuristic that scales the contribution of the patterns appropriately is admissible. To elaborate this further we need some additional definitions.

Definition 14. *Two patterns P_1 and P_2* conflict *if there is a cost component C that exists in both $\mathcal{T}_{P_1}$ and $\mathcal{T}_{P_2}$.*

We will use the notion of conflicts to argue why pattern collections are additive. We do so by reasoning about sequences that are shared between patterns, which is simpler than reasoning about the occurrence of cost components.

Proposition 3. *P_1 and P_2 conflict iff $|P_1 \cap P_2| \geq 2$.*

Proof. If $|P_1 \cap P_2| \geq 2$, then P_1 and P_2 have at least two sequences s_1 and s_2 in common. Thus, there exist cost components of the form $C^{1,2}_{\langle x,y \rangle \to \langle x'y' \rangle}$ that label transitions in both projections, so P_1 and P_2 are in conflict. For the other direction, observe that if there exists a cost component C that appears in both $\mathcal{T}_{P_1}$ and $\mathcal{T}_{P_2}$, then by definition these two patterns have at least two sequences in common. □

We extend the notion of conflicts to pattern collections.

Definition 15. *A pattern collection $\mathcal{P}$ is* conflicting *if there exist two patterns $P_i, P_j \in \mathcal{P}$ that conflict, with $P_i \neq P_j$.*

With this we have a criterion for admissibility based on conflicts.

Proposition 4. *Let $\mathcal{P}$ be a pattern collection. If $\mathcal{P}$ is not conflicting, then $h^{\mathcal{P}}$ is admissible.*

Proof. As $\mathcal{P}$ is not conflicting, no cost component C exists in more than one projection $\mathcal{T}_{P_i}$ induced by the patterns $P_i \in \mathcal{P}$. Thus, over all solutions π_i for all $\mathcal{T}_{P_i}$, each cost component is counted at most once. □

We also introduce a notion for when all patterns in the pattern collection have two sequences in common.

Definition 16 (Strictly conflicting MSA pattern collections)**.** *Let $\mathcal{P}$ be a pattern collection. $\mathcal{P}$ is* strictly conflicting *iff $|\bigcap_{P \in \mathcal{P}} P| \geq 2$.*

This allows us to consider minimal pattern sub-sets $\mathcal{P}' \subseteq \mathcal{P}$ that all share at least two sequences, which relates to the number of shared occurrences of cost components.

Uniform Cost Partitioning over MSA Abstractions

We redefine the UCP algorithm to a version for MSA abstractions.

Definition 17. *(MSA UCP) The MSA UCP algorithm distributes each cost component C over patterns $\mathcal{P} = \{P_1, \ldots, P_n\}$ by using the fractional cost components $w_{P_1}(C), \ldots, w_{P_n}(C)$ instead, where*

$$w_P(C) = \begin{cases} \frac{1}{|\{P \in \mathcal{P} | C \text{ exists in } \mathcal{T}_P\}|} & \text{if } C \text{ exists in } \mathcal{T}_P \\ 0 & \text{otherwise.} \end{cases}$$

The resulting MSA UCP heuristic is defined as

$$h^{\mathcal{P}}_{\text{UCP}}(s) = \sum_{P \in \mathcal{P}} h^{P'}(s_{|P}),$$

where $h^{P'}$ is the heuristic for pattern P, but where the underlying transition system $\mathcal{T}_P$ has its label costs recomputed to use cost $w_P(C)$ for each cost component C.

Theorem 5. *$h^{\mathcal{P}}_{\text{UCP}}$ is admissible.*

Proof. For the given pattern collection $\mathcal{P} = \{P_1, \ldots, P_n\}$, each cost component C is distributed over patterns $P_1, \ldots P_n$ as $w_{P_1}(C), \ldots, w_{P_n}(C)$, so that $\sum_{i=1,\ldots,n} w_{P_i}(C) = 1$. Since any solution for every projection $\mathcal{T}_{P_i}$ uses C at most once (Proposition 1), C never contributes more than its original costs. □

From now on, we use the notation $h^{\text{all},k}_{\text{UCP}}$ and $h^{\text{one},k}_{\text{UCP}}$ to refer to $h^{\mathcal{P}}_{\text{UCP}}$ where the pattern collection $\mathcal{P}$ corresponds to the patterns considered by the $h^{\text{all},k}$ and $h^{\text{one},k}$ heuristics, respectively. We will use the same notation for $h^{\mathcal{P}}_{\text{PhO}}$, introduced below.

We now show that $h^{\mathcal{P}}_{\text{UCP}}$ is equal to $h^{\text{all},k}$ and $h^{\text{one},k}$, for a suitable choice of patterns.

Proposition 6. $h^{\text{one},k}_{\text{UCP}} = h^{\text{one},k}$

Proof. Given sequences $S = s_1, \ldots, s_n$, some value $2 \geq k \geq n$, and a pattern collection $\mathcal{P}$ consisting of $P_1 \subseteq S$ where $|P_1| = 2$, $P_2 = S/P_1$, and all patterns in $\mathcal{P}' = \bigcup_{s_i \in P_1, s_j \in P_2} \{s_i, s_j\}$, we have that for every cost component C, it exists in at most one pattern P of $\mathcal{P}$ because $\mathcal{P}$ is non-conflicting. Consequentially, the contribution for C is 1, leading to the full heuristic value per pattern, which is equal to $h^{\text{one},k}$. □

Proposition 7. $h^{\text{all},k}_{\text{UCP}} = h^{\text{all},k}$

Proof. Given sequences $S = s_1, \ldots, s_n$, some value $2 \geq k \geq n$, and any state s, then we have a pattern collection $\mathcal{P}$ of all k-folds. Each cost component of the form $C^{i,j}$ where $1 \geq i < j \geq n$ then exist in exactly $\binom{n-2}{k-2}$ patterns. As a result, every cost component in each pattern P_i where $i = 1, \ldots \binom{n}{k}$ will contribute $\frac{1}{\binom{n-2}{k-2}}$ of its associated cost, therefore we can rewrite the heuristic value per pattern as $\frac{h^{P_i}(s)}{\binom{n-2}{k-2}}$. The heuristic value $h^{\text{all},k}_{\text{UCP}}(s)$ is then

$$\sum_{i=1}^{\binom{n}{k}} \frac{h^{P_i}(s)}{\binom{n-2}{k-2}} = \frac{1}{\binom{n-2}{k-2}} \sum_{i=1}^{\binom{n}{k}} h^{P_i}(s) = h^{\text{all},k}(s). \qquad \square$$

Post-hoc Optimization over MSA Abstractions

We define the MSA PhO LP for a given pattern collection $\mathcal{P} = \{P_1, \ldots, P_n\}$ and state s:

$$h^{\mathcal{P}}_{\text{PhO}}(s) = \text{maximize} \sum_{i=1}^{n} w_i \cdot h^{P_i}(s) \text{ s.t.}$$
$$\sum_{P_i \in \mathcal{P}'} w_i \leq 1 \text{ for all strictly conflicting } \mathcal{P}' \subseteq \mathcal{P}$$
$$w_i \geq 0 \text{ for all } P_i \in \mathcal{P}.$$

Theorem 8. *$h^{\mathcal{P}}_{\text{PhO}}$ is admissible.*

Proof. Each cost component C exists in all patterns P_i of a strictly conflicting pattern collection $\mathcal{P}$ and C does not exist in any pattern from $\mathcal{P} \setminus \mathcal{P}'$. Since $\sum_{P_i \in \mathcal{P}'} w_i \leq 1$, we therefore also get $\sum_{P_i \in \mathcal{P}} w_i(C) \leq 1$. Consequentially, the constraints of the MSA PhO LP ensure that the contribution of each cost component C does not exceed its original cost. □

We now have the tools to formally relate the $h^{\text{all},k}$ heuristic to post-hoc optimization. We start by proving that $h^{\text{all},k}$ is a cost partitioning heuristic, and that it is dominated by post-hoc optimization.

Theorem 9. *$h^{\text{all},k}_{\text{PhO}}$ dominates $h^{\text{all},k}$.*

Proof. Consider the pattern collection $\mathcal{P} = \{P_1, \ldots, P_{\binom{n}{k}}\}$ consisting of all $\binom{n}{k}$ k-fold patterns from a set of n sequences S. Then we can compute $h^{\text{all},k}(s)$ for a state s as

$$h^{\text{all},k}(s) = \frac{\sum_{i=1}^{\binom{n}{k}} h^{P_i}(s)}{\binom{n-2}{k-2}} = \sum_{i=1}^{\binom{n}{k}} \frac{1}{\binom{n-2}{k-2}} h^{P_i}(s)$$

by using the definition of $h^{\text{all},k}$ and simple arithmetic. After setting $w_i = \frac{1}{\binom{n-2}{k-2}}$, it becomes apparent that the result, $h^{\text{all},k}(s) = \sum_{i=1}^{\binom{n}{k}} w_i \cdot h^{P_i}(s)$ is a valid solution to the MSA PhO LP. To see that $h^{\text{all},k}(s)$ satisfies both LP constraints, observe that

(i) Each pair of sequences appears in exactly $\binom{n-2}{k-2}$ patterns. Therefore, the number of strictly conflicting patterns for each sequence is $\binom{n-2}{k-2}$ and we have $\sum_{P_i \in \mathcal{P}'} w_i = \binom{n-2}{k-2} \frac{1}{\binom{n-2}{k-2}} = 1$, and

(ii) $0 < \binom{n-2}{k-2} \Rightarrow \frac{1}{\binom{n-2}{k-2}} \geq 0$ □

By the above observations we conclude that the factor $\frac{1}{\binom{n-2}{k-2}}$ is a valid value for all w_i.

There are even states in MSA tasks where $h^{\text{all},k}_{\text{PhO}}$ is strictly more accurate than $h^{\text{all},k}$.

Proposition 10. *The dominance relation between $h^{\text{all},k}_{\text{PhO}}$ and $h^{\text{all},k}$ is strict.*

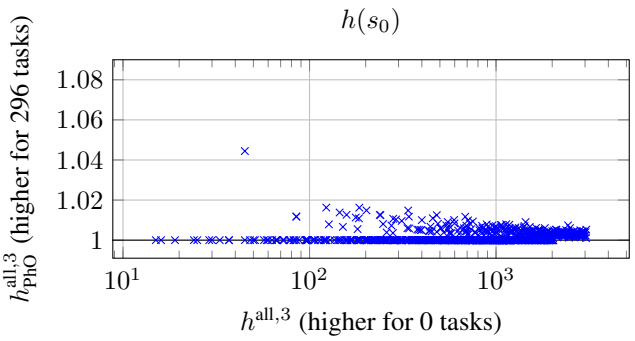

Figure 6: Per-instance ratio of initial-state heuristic value of $h^{\text{all},3}_{\text{PhO}}$ over $h^{\text{all},k}$, as a function of the absolute value of $h^{\text{all},k}$.

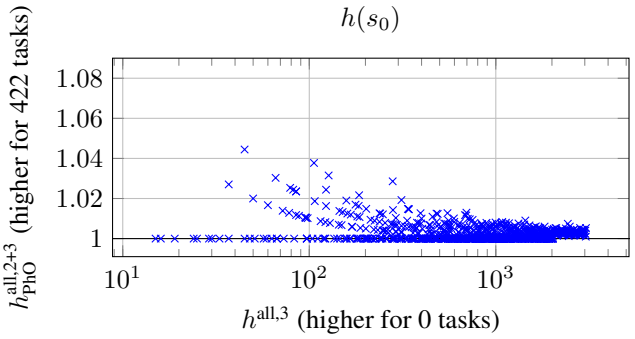

Figure 7: Per-instance ratio of initial-state heuristic value of $h^{\text{all},2+3}_{\text{PhO}}$ over $h^{\text{all},k}$, as a function of the absolute value of $h^{\text{all},k}$.

Proof. Consider the following MSA task with five sequences and the pattern collection consisting of all $\binom{5}{3} = 10$ 3-folds $\mathcal{P} = \{P_1, \ldots, P_{10}\}$. Let the 10 heuristic estimates for state s be $\langle 12, 12, 12, 6, 10, 10, 6, 10, 10, 4\rangle$. Then $h^{\text{all},k}(s) = \frac{12+12+12+6+10+10+6+10+10+4}{3} = \frac{92}{3} < 32 = (12 \cdot 0.5 + 12 \cdot 0.5 + 12 \cdot 0 + 6 \cdot 0 + 10 \cdot 0.5 + 10 \cdot 0.5 + 6 \cdot 0 + 10 \cdot 0.5 + 10 \cdot 0.5 + 4 \cdot 0) = h^{PhO}(s)$.[1] □

In experiments we verify that this strict relationship holds for BaliBase tasks. Next, we observe that MSA PhO is comparable to $h^{\text{one},k}$.

Theorem 11. $h^{\text{one},k}_{\text{PhO}} = h^{\text{one},k}$.

Proof. The pattern collection $\mathcal{P}$ is non-conflicting, therefore the optimal solution to the LP will set $w_i = 1$ for $i = 1, \ldots, n$. Thus the heuristic value will be equal to that of $h^{\text{one},k}$. □

Experiments

We integrated our new cost-partitioning heuristics into MSASolver,[2] a specialized Java implementation of A* search for MSA problems that includes heuristics such as $h^{\text{all},k}$ and $h^{\text{one},k}$. We extend the code from Riesterer (2018), which integrated post-hoc optimization into MSASolver. Our main goal for the evaluation is to confirm our theoretical results on the dominance of post-hoc optimization over $h^{\text{all},k}$. Hence, we focus on initial-state heuristic values throughout our analysis. To conduct the experiments, we use the Lab Python package (Seipp et al. 2017). Each task is run on an AMD Ryzen 7 PRO 5850U CPU, with 30 minute runtime and 20 GiB memory limits. We evaluate both heuristics on two benchmark sets, which we describe next.

BAliBase

The BAliBase benchmark collection contains multiple sets of instances of protein sequences (Thompson, Plewniak, and Poch 1999). We use the full benchmark set consisting of 898 instances, each with 4–419 sequences and up to 1382 characters. In our evaluation, we obtained results for all instances with at most six sequences, which is a general restriction of MSASolver. Out of the full benchmark set, 128 instances satisfy this condition.

Random DNA Sequences

We increase the size of our benchmark set to enable a more systematic evaluation by generating 900 instances consisting of random DNA sequences. These contain the nucleotide characters A,C,G and T. For every number of 4–6 sequences and lengths from 1–100 characters, we generate three random character sequences. Besides increasing the size of our benchmark set, the pairwise substitution costs in DNA sequences is very different from the costs in the protein sequences of BAliBase, so we obtain more diverse benchmarks as well.

Results

We have shown above that $h^{\text{all},k}_{\text{PhO}}$ strictly dominates $h^{\text{all},k}$. Here, we want to experimentally confirm this result by comparing the two heuristics on both benchmark sets. Our evaluation considers two settings: first, we compare both approaches when using the same set of patterns, namely all projections of size three. Besides this, we show results for $h^{\text{all},3}$ in relation to post-hoc optimization with all patterns of sizes two and three, which we denote by $h^{\text{all},2+3}_{\text{PhO}}$. The latter exemplifies the flexibility of post-hoc optimization, which, in contrast to $h^{\text{all},k}$, supports arbitrary pattern collections.

The observations are similar on both benchmark sets. We see that $h^{\text{all},3}_{\text{PhO}}$ yields a higher heuristic value than $h^{\text{all},3}$ frequently, with an increase of up to 5% when considering the DNA sequences. On the BAliBase instances, the advantage of post-hoc optimization is smaller, with a maximum improvement of 0.07%. As expected, $h^{\text{all},3}$ never yields higher estimates than $h^{\text{all},3}_{\text{PhO}}$.

The plots in Figures 6 and 7 show the improvement of post-hoc optimization over $h^{\text{all},3}$ on the DNA sequences, where the different heuristic values can be nicely visualized. Each point in a plot represents one MSA instance, where the x-value is the heuristic value of $h^{\text{all},3}$ and the y-value is

[1] This is the smallest BAliBase benchmark instance we could find to show this strict dominance.

[2] MSASolver is available at https://github.com/matthatem/MSASolverhttps://github.com/matthatem/MSASolver.

the ratio of $h^{\text{all},3}_{\text{PhO}}$ over $h^{\text{all},3}$. So values greater than $y = 1$ indicate that $h^{\text{all},3}_{\text{PhO}}$ obtains a higher heuristic value.

In Figure 6, we compare both heuristics on the same pattern collection that consists of all projections of size three. Here, $h^{\text{all},3}_{\text{PhO}}$ yields a higher heuristic value in 296 out of the 900 instances. In Figure 7 we highlight that post-hoc optimization can be computed over arbitrary pattern collections. We observe that this flexibility indeed pays off and the heuristic improves over $h^{\text{all},k}$ in 422 instances. This indicates the potential of the more versatile cost partitioning methods. For future work, we hypothesize that the pattern selection can be tailored for specific cost partitioning methods, so that higher heuristic values can be achieved, e.g., by finding a good middle ground between the collections used in $h^{\text{all},k}$ and $h^{\text{one},k}$.

Conclusions

We have established novel theoretical connections between existing heuristics for MSA and cost partitioning. We introduced cost components to reason about the transition costs of specific interactions between two sequences. Using this concept, we adapted uniform cost partitioning and post-hoc optimization, two well-known cost-partitioning methods from automated planning, and developed two new heuristics h_{UCP} and h_{PhO} respectively. We showed that these two heuristics are admissible and established that, for pattern collections consisting of all k-folds, h_{UCP} is $h^{\text{all},k}$ and that h_{PhO} strictly dominates $h^{\text{all},k}$. Furthermore, we showed that both heuristics dominate $h^{\text{one},k}$ if using the same pattern collection as $h^{\text{one},k}$. In our experimental evaluation, we verified that the dominance relationship between $h^{\text{all},k}$ and h_{PhO} is not merely a theoretical curiosity, but that these differences do frequently have an impact on the initial heuristic value. Finally, our experiments also highlighted that the flexibility of h_{PhO} allows the heuristic to yield even higher heuristic values in many instances, when not restricted to the pattern collections of $h^{\text{all},k}$.

For future work, we will investigate by what factor this difference in initial heuristic value, and differences tied to the cost partitioning algorithms themselves, carry over to the actual search with A*. This includes other techniques applied during search, too, such as deciding when to recompute cost partitionings, which is necessary for post-hoc optimization (Höft, Speck, and Seipp 2023). We also want to investigate more cost-partitioned heuristics for MSA, such as saturated cost partitioning (Seipp, Keller, and Helmert 2017) and *saturated* post-hoc optimization (Seipp, Keller, and Helmert 2021), as well as preprocessing techniques for pattern collections. For instance, can we show that a pattern collection $\mathcal{P}$ will never achieve a higher heuristic value than a collection $\mathcal{P}'$ when employing a specific cost-partitioning method? In particular, we have already found that for four sequences, we can disregard up to 80% of the candidate pattern collections that are subsets of all 2-folds and 3-folds. The question is whether this can be generalized to any number of sequences.

References

Carrillo, H.; and Lipman, D. 1988. The Multiple Sequence Alignment Problem in Biology. *SIAM Journal on Applied Mathematics*, 48(5): 1073–1082.

Culberson, J. C.; and Schaeffer, J. 1998. Pattern Databases. *Computational Intelligence*, 14(3): 318–334.

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

Felner, A.; Korf, R.; and Hanan, S. 2004. Additive Pattern Database Heuristics. *JAIR*, 22: 279–318.

Franco, S.; Torralba, Á.; Lelis, L. H. S.; and Barley, M. 2017. On Creating Complementary Pattern Databases. In *Proc. IJCAI 2017*, 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.; 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.

Höft, P.; Speck, D.; and Seipp, J. 2023. Sensitivity Analysis for Saturated Post-hoc Optimization in Classical Planning. In *Proc. ECAI 2023*, 1044–1051.

Ikeda, T.; and Imai, H. 1994. Fast A* Algorithms for Multiple Sequence Alignment. *Genome Informatics*, 5: 90–99.

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.

Katz, M.; and Domshlak, C. 2010. Optimal admissible composition of abstraction heuristics. *AIJ*, 174(12–13): 767–798.

Kobayashi, H.; and Imai, H. 1998. Improvement of the A* Algorithm for Multiple Sequence Alignment. *Genome Informatics*, 9: 120–130.

Needleman, S. B.; and Wunsch, C. D. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3): 443–453.

Pearl, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015. From Non-Negative to General Operator Cost Partitioning. In *Proc. AAAI 2015*, 3335–3341.

Pommerening, F.; Keller, T.; Halasi, V.; Seipp, J.; Sievers, S.; and Helmert, M. 2021. Dantzig-Wolfe Decomposition for Cost Partitioning. In *Proc. ICAPS 2021*, 271–280.

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

Riesterer, M. 2018. *Cost Partitioning Techniques for Multiple Sequence Alignment*. Master's thesis, University of Basel.

Seipp, J.; and Helmert, M. 2014. Diverse and Additive Cartesian Abstraction Heuristics. In *Proc. ICAPS 2014*, 289–297.

Seipp, J.; and Helmert, M. 2018. Counterexample-Guided Cartesian Abstraction Refinement for Classical Planning. *JAIR*, 62: 535–577.

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.; Keller, T.; and Helmert, M. 2020. Saturated Cost Partitioning for Optimal Classical Planning. *JAIR*, 67: 129–167.

Seipp, J.; Keller, T.; and Helmert, M. 2021. Saturated Post-hoc Optimization for Classical Planning. In *Proc. AAAI 2021*, 11947–11953.

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

Thompson, J. D.; Plewniak, F.; and Poch, O. 1999. BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, 15(1): 87–88.

Yang, F.; Culberson, J.; Holte, R.; Zahavi, U.; and Felner, A. 2008. A General Theory of Additive State Space Abstractions. *JAIR*, 32: 631–662.