A Comparison of Cost Partitioning Algorithms for Multiple Sequence Alignment

Mika Skjelnes Daniel Gnad Jendrik Seipp

Linköping University · Heidelberg University

HSDIP 2026

Multiple Sequence Alignment

sequences

s_1:	G
s_2:	T
s_3:	T

$\Rightarrow$

an alignment

	col_1	col_2
A_1:	−	G
A_2:	T	−
A_3:	−	T

boxed col_1 : $cost(col_1) = sub(-, T) + sub(T, -) = 3 + 3 = 6$

pairwise substitution cost (ACGT)

	A	C	G	T	−
A	0	4	2	2	3
C	4	1	3	4	3
G	2	3	1	6	3
T	2	4	6	0	3
−	3	3	3	3	

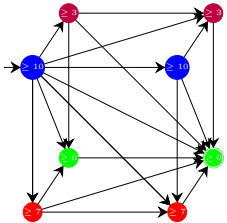
- Given k sequences, insert gaps so all sequences have equal length.
- Goal: **minimize the alignment cost**.

Solving MSA as Heuristic Search

- Solve for a subset $P \subseteq Q$ of sequences.
- This is essentially a **pattern database** (PDB).

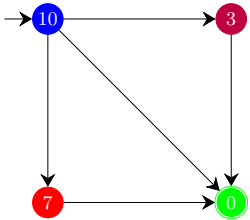
How to combine multiple PDBs?

Cost partitioning doesn't work out-of-the-box.



$\Rightarrow$

project onto a pair

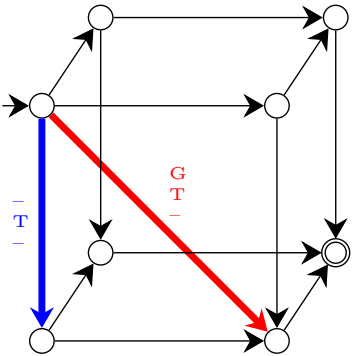


Cost Structure of MSA: Factored Transition Costs

- Cost of a transition t is the cost of the corresponding column in the alignment.
- Cost of a column = sum of its pairwise substitution costs (cost components).
- $\Rightarrow$ cost of a transition = sum of its **cost components**

Why it matters

It enables **cost partitioning**!



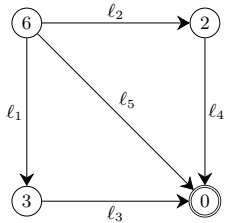
blue edge = column $(-, T, -)$, red edge = column $(G, T, -)$;
each edge's cost = sum of its cost components

- Blue edge: $\{C^{1,2}_{\langle 0,0\rangle \to \langle 0,1\rangle}, C^{2,3}_{\langle 0,0\rangle \to \langle 1,0\rangle}\}$
- Red edge: $\{C^{1,2}_{\langle 0,0\rangle \to \langle 1,1\rangle}, C^{1,3}_{\langle 0,0\rangle \to \langle 1,0\rangle}, C^{2,3}_{\langle 0,0\rangle \to \langle 1,0\rangle}\}$

Cost Saturation & Redistribution

With cost components we get a redistribution problem, unlike in *textbook cost saturation*.

$$\sum_{c_i \in \ell} rem(c_i) \leq rem(\ell)$$



$cost(\ell_1) = 3$
$cost(\ell_2) = 4$
$cost(\ell_3) = 3$
$cost(\ell_4) = 2$
$cost(\ell_5) = 10$

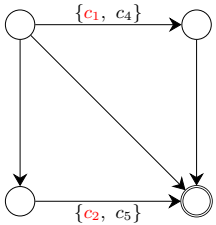
$$\ell_5 = \{c_1, c_2, c_3\}$$

$$x_1 + x_2 + x_3 \leq 4$$

Result 1: Greedy Redistribution - Two Realizations

1. Components don't compete

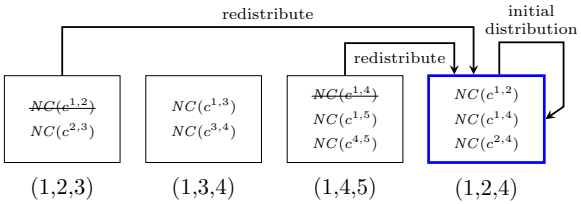
If all $|P| \leq 3$ then two cost-components in a remaining-cost constraint never co-occcur in another pattern



$$x_1 + x_2 + x_3 \leq 4$$

2. Only the most recent PDB matters

Only look at the most recent PDB containing remaining cost constraints for a cost component c; $NC(c)$



$\Rightarrow$ greedy redistribution: Trivial optimization per cost component c, and only check remaining cost constraints of NC(c)

Result 2: $h^{\text{PhO}} = h^{\text{SPhO}}$

In classical planning, $h^{\text{SPhO}} \geq h^{\text{PhO}}$. In MSA, the two LPs coincide:

h^{PhO}

$$\text{maximize} \sum_{h \in \mathcal{H}} h(cost, s) \cdot w_h \text{ s.t.}$$

$$\sum_{h \in \mathcal{H}} cost(c)\, w_h \leq cost(c) \text{ for all cost components } c$$

$$w_h \geq 0 \quad \text{for all } h \in \mathcal{H}$$

h^{SPhO}

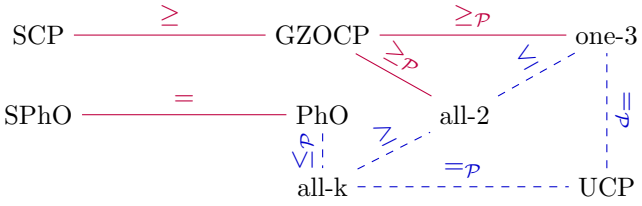
$$\text{maximize} \sum_{h \in \mathcal{H}} h(cost, s) \cdot w_h \text{ s.t.}$$

$$\sum_{h \in \mathcal{H}} scf_h(c)\, w_h \leq cost(c) \text{ for all cost components } c$$

$$w_h \geq 0 \quad \text{for all } h \in \mathcal{H}$$

Key: Every cost component $c \in \mathcal{C}_{ij}$ exists in the same set of PDBs.

Result 3: Every Baseline is Dominated



purple = new (this work), blue dashed = known, $\mathcal{P}$ = under a pattern collection

- Every MSA-specific heuristic is dominated by a cost partitioning algorithm (same patterns).
- But h^{GZOCP}/h^{SCP} don't dominate $h^{\text{all-3}}$; only per-state h^{PhO} does (needs the LP).

Experiments: Pairwise Comparison (Easy BAliBASE, 75 tasks)

Expansions below $C^\star$ (tie-break independent)

		h^{GZOCP}		h^{SCP}	h^{PhO}			Baselines			
		2D-first	best2-first	best2-first	$K{=}0$	$K{=}1$	$K{=}10k$	$h^{\text{one-3}}$	$h^{\text{all-2}}$	$h^{\text{all-3}}$	Solved
h^{GZOCP}	2D-first	–	0	0	0	0	0	0	0	0	94.7%
	best2-first	69	–	0	0	0	0	26	69	4	92.0%
h^{SCP}	best2-first	52	47	–	2	1	2	47	52	6	70.7%
h^{PhO}	$K{=}0$	67	63	48	–	2	0	63	67	20	92.0%
	$K{=}1$	67	64	49	55	–	52	64	67	57	90.7%
	$K{=}10k$	67	63	48	5	5	–	63	67	24	92.0%
Baselines	$h^{\text{one-3}}$	70	0	0	0	0	0	–	70	2	93.3%
	$h^{\text{all-2}}$	0	0	0	0	0	0	0	–	0	94.7%
	$h^{\text{all-3}}$	67	59	44	9	1	5	62	67	–	92.0%

Total time

		h^{GZOCP}		h^{SCP}	h^{PhO}			Baselines			
		2D-first	best2-first	best2-first	$K{=}0$	$K{=}1$	$K{=}10k$	$h^{\text{one-3}}$	$h^{\text{all-2}}$	$h^{\text{all-3}}$	Solved
h^{GZOCP}	2D-first	–	49	46	42	46	42	49	25	42	94.7%
	best2-first	20	–	52	7	14	5	18	19	6	92.0%
h^{SCP}	best2-first	6	0	–	0	0	0	2	6	0	70.7%
h^{PhO}	$K{=}0$	25	60	53	–	41	27	49	23	15	92.0%
	$K{=}1$	21	53	53	27	–	18	46	20	14	90.7%
	$K{=}10k$	25	62	53	42	49	–	51	23	19	92.0%
Baselines	$h^{\text{one-3}}$	21	50	50	18	20	16	–	18	10	93.3%
	$h^{\text{all-2}}$	45	50	46	44	47	44	51	–	44	94.7%
	$h^{\text{all-3}}$	25	61	53	53	52	48	57	23	–	92.0%

Cell (i, j) = #tasks both solve where i is **better** than j (fewer $f < C^\star$ expansions / faster). Last column = coverage.

- $h^{\text{PhO}} \succeq h^{\text{all-}k}$, and $h^{\text{SCP}} \succeq h^{\text{GZOCP}}$, but in both cases the overhead of the stronger heuristic doesn't pay off.[1]

[1] Per-state h^{PhO} shows $\leq 0.25\%$ more $f < C^\star$ on 2/5 tasks; a floating-point boundary effect

Discussion: Beating all-3, and Who Benefits

- **Can we beat $h^{\text{all-}k}$?** Its strength is minimum overhead; CP's lever is **flexibility**; choose which patterns to materialize (filtering).
- Perhaps BAliBASE tasks with more than 6 sequences can change the story?
- **Biology?** Heuristic-search MSA is detached from practice, but the *factored cost structure* is general (shared costs split among exactly-solved subproblems).

Where can factored-cost cost partitioning travel?

Takeaways

Cost partitioning provides a promising toolkit for MSA, but it might require pattern engineering to beat the best baselines.

Factored costs make redistribution *ambiguous*; **greedy** redistribution makes it cheap and pattern-order-dependent. Doesn't seem to pay off currently.

Any thoughts or suggestions?

mika.skjelnes@liu.se

Backup: ACGT many-sequence regime (30-min timeout)

8 hard ACGT tasks, **15–17 sequences** — short sequences, but a ∼6000-way branching factor.

Coverage: h^{PhO} 5/8 $>$ h^{SCP} 3/8 $>$ $h^{\text{all-3}}$ 1/8 ($h^{\text{one-3}}$, h^{GZOCP}, filtered h^{SCP}: 0/8)

task	h^{PhO} (K=0) exp / time	h^{SCP} (best2) exp / time	$h^{\text{all-3}}$ exp / time
ins_15_0	2887 / 588 s	–	–
ins_15_1	4111 / 717 s	7893 / 1002 s	–
ins_16_0	1053 / 499 s	2020 / 746 s	4354 / 1211 s
ins_16_2	2260 / 945 s	4158 / 1385 s	–
ins_17_2	282 / 322 s	–	–

Solved tasks expand only ∼1–8k nodes but **generate 5–20M** ⇒ search is heuristic-bound; a sharper h prunes the fan-out and wins. Per-node strength becomes coverage: $h^{\text{PhO}} \succ h^{\text{SCP}} \succ h^{\text{all-3}}$. `ins_16_1/17_0/17_1` unsolved by all; one M/O ($h^{\text{one-3}}$ on `ins_17_2`).