On Solving the Multiple Variable Gapped Longest Common Subsequence Problem

Marko Djukanović[1,2, 6] Nikola Balaban[2] Christian Blum[3]
Aleksandar Kartelj[4] Sašo Džeroski[5] Žiga Zebec[6]

[1]University of Nova Gorica, Nova Gorica, Slovenia
[2]Faculty of Natural Sciences and Mathematics, University of Banja Luka, Banja Luka, B&H
[3]Artificial Intelligence Research Institute (IIIA-CSIC), Barcelona, Spain
[4]Faculty of Mathematics, University of Belgrade, Belgrade, Serbia
[5]Jožef Stefan Institute, Ljubljana, Slovenia
[6]Institute of Information Sciences (IZUM), Maribor, Slovenia

– EUROCAST 2026: 20th International Conference on Computer Aided Systems Theory, February 23-27, 2026, Las Palmas de Gran Canaria, Spain–

Outline

- Introduction
- Problem Definition
- Rooted graph state space
- Iterative multi-source Beam Search
- Experimental Evaluation: General & special problem
- Conclusions

Introduction

- Objects we deal with: sequences (strings) over finite alphabet Σ
  - DNA/RNA over $\{\mathtt{A}, \mathtt{T}, \mathtt{G}, \mathtt{C/U}\}$
  - Proteins over 20 (canonical) amino acids: $\{\mathtt{A}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{P}, \mathtt{Q}\ldots\}$
- **One of central tasks in computational biology:**
  - sequence comparison, finding common motifs between sequences
  - compare structurally but also semantically/functionality
  - sequence alignment problems
- **Subsequences**: reveal structural similarities $\rightarrow$ **Longest commmon subsequence problem** variants (studied 50 years already)

Longest common subsequence problem (LCSP)

Definition (LCSP)

Input: Given an arbitrary set of sequences $S = \{s_1, \ldots, s_m\}$
Task: Find a subsequence s common for all sequences from S with maximum possible length ($|s|$).

Example

Input: $S = \{$`AATTGC`, `ATTAC`$\}$
LCS solution: $s =$ATTC

- Basic problem in computational biology, well-solved theoretically and practically

LCS: Literature & Problem Variants

- When $m = 2$ – polynomially solvable (in $O(n^2)$): Dynamic programming, Hunt-Szymanski, Hirschenberg, ...
- When m arbitrary large – $\mathcal{NP}$-hard:
    - subject of interest within last 30 years: approximation approaches, meta-heuristics (ACO, Beam search, ...), but also exact approaches (A*, anytime approaches, DAG-based, ...)

Problem-related practical variants:

- Arc-annotated LCS problem
- Constrained, Repetition-free, filled LCS problem, ...
- Gapped LCS problem

The gapped LCS problem

Definition (A gap sequence)

Given is a sequence s and an assigned function $G_s\colon \{1, \ldots, |s|\} \mapsto \mathbb{N}$. A pair (s, G_s) is called a **sequence with gaps**.

Definition (A gapped subsequence)

Sequence $\tilde{s}$ is a gapped subsequence of (s, G_s) iff

- $\tilde{s}$ is a subsequence of s
- the gap constraint G_s is fulfilled w.r.t. *positions of appearances* of letters of $\tilde{s}$ in s
  - suppose $i_1, \ldots, i_{|\tilde{s}|}$ are **positions of embedding** $\tilde{s}$ in s
  - $(\forall\ j = 2, \ldots, |\tilde{s}|)\ i_j - i_{j-1} \leq G_s(i_j) + 1$

Problem definition

Example

$s =$ `AATTGC`, $G_s(\cdot) = 1$

- $\tilde{s} =$ `ATG`, the embedding: AATTGC (valid gapped subsequence)
- $\tilde{s} =$ `ATG`, the embedding: AATTGC (invalid gapped subsequence)

Definition (The multiple (variable) gapped LCS problem – MVGLCSP)

Input: Given is a set of gapped sequences $\{(s_1, G_{s_1}), \ldots, (s_m, G_{s_m})\}$.
Task: Find the longest subsequence $\tilde{s}$ so that

- $\tilde{s}$ is common subusequence of each s_i
- $\tilde{s}$ fulfills all gap constraints G_{s_i} $(i = 1, \ldots, m)$

Note: when $G_{s_i} = n$ (the length of longest sequence) $\Rightarrow$ VGLCSP $=$ LCSP.

Literature & Motivation for VGLCSP

- Peng and Yang (2012, 2014): studied the $m = 2$ (poly) version by **three dynamic programming** approaches (basic one, two advanced involving complex data-structures to speed up)
- Manea et al. (2024): Complexity bounds, (parameterised) complexity analysis investigated
- NP-hard under arbitrary large m

Motivation:

- Genetics and molecular biology: applications in DNA/protein analysis where variable structural distances between residues must be respected
- Time-series analysis: in settings where events are required to occur within specified temporal delays (Lainscsek et al. (2015))

Methodology

- Based on the significant extension of the state space graph formulation for LCS problem (Djukanovic et al. (2020))
- **Gap constraints**: incorporated to cut-off invalid extensions (edges) among the LCS extensions immediately
- Many root/source nodes in the state graph, generally separated subgraphs (the leading letter in a common subsequence free)

Root-based state graph formulation: rough idea

- Each **state** $v = (p^L, l^v)$: one or more feasible partial solutions where
  - a vector of positions p^L refer to the positions of suffixes of input sequences relevant to further expand these sols
  - the length of current partial solution l^v

- **Expansion** of v: extend (concatenate) partial solutions feasibly (and non-dominantly) by one letter in all possible ways, respecting gap constraints

- **Non-expandable nodes**: complete solutions;

Decision: select appropriate (match) p^L for start $\Rightarrow$ possibly **(exponentially) many root nodes**

Space(p^L): root-based state (sub) graph induced by node $(p^L, 0)$.

Root-based state graph formulation: example with the match $r = (1, 1)$ (obviously a valid root node)

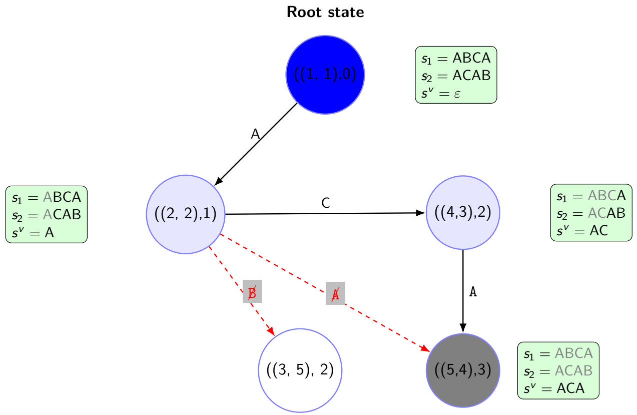


Figure: State space graph Space($r = ((1, 1), 0)$) for MVGLCSP between the sequences $\boxed{\texttt{A}}$`BCA` and $\boxed{\texttt{A}}$`CAB`, assuming $G_1 = G_2 = 1$.

An issue with the root-state-space formulation: example

Example

$S = \{s_1 = \texttt{ATGG}\boxed{\texttt{A}}\texttt{AA},\ s_2 = \texttt{ATCC}\boxed{\texttt{A}}\texttt{AA}\}$, with gap constraints $G_{s_1} = G_{s_2} = 1$. In this instance, any state with position vector $\mathbf{p}^L = (5,5)$ cannot be reached from the initial state $((1,1),0)$ by standard direct transitions.

Consequence: $((5,5),0) \notin \mathit{Space}((1,1)) \Rightarrow$ The optimal common subsequence `AAA` is unreachable

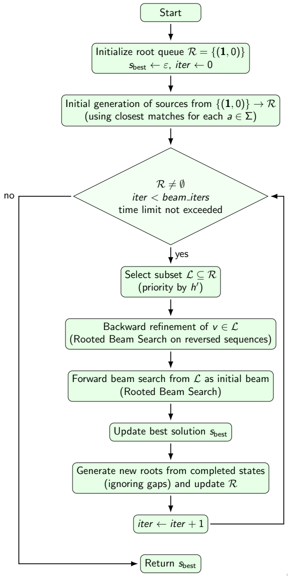
How to fix it?

Iterative multi-source beam search (IMSBS) strategy

- Explicitly enumerating all root states computationally prohibitive ($O(n^m)$ time)

- **IMSBS** (rough idea):
  - Exploring Space(r): **beam search** — limited breadth-first-search (BFS) strategy; parameter β controls the number of nodes to be further pursued
  - Dynamically explore multiple promising regions (Space($\cdot$)) of the state space
  - **Iteratively identify** a set of promising candidate root states

Workflow of the IMSBS


Start
Initialize root queue $\mathcal{R} = \{(\mathbf{1}, 0)\}$
$s_{\text{best}} \leftarrow \varepsilon$, $iter \leftarrow 0$
Initial generation of sources from $\{(\mathbf{1}, 0)\} \rightarrow \mathcal{R}$
(using closest matches for each $a \in \Sigma$)
$\mathcal{R} \neq \emptyset$
$iter < beam\_iters$
time limit not exceeded
no
yes
Select subset $\mathcal{L} \subseteq \mathcal{R}$
(priority by h')
Backward refinement of $v \in \mathcal{L}$
(Rooted Beam Search on reversed sequences)
Forward beam search from $\mathcal{L}$ as initial beam
(Rooted Beam Search)
Update best solution s_{best}
Generate new roots from completed states
(ignoring gaps) and update $\mathcal{R}$
$iter \leftarrow iter + 1$
Return s_{best}

Heuristic guidances in BS

Three LCS heuristic guidances used:

- “Look-ahead” for the remaining sequence length: $\mathrm{UB}_1(v)$
- *Character Frequency Alignment* score: UB_2
  - sum up the maximum possible occurrences of each letters in subsequences
- *A probability-based heuristic* guidance: h_{prob} (the pre-processed matrices of probabilities (Mousavi and Tabataba (2012)))

Experimental Evaluation: arbitrary large *m*-case

- BS: a baseline beam search approach, allowing only a single iteration of IMSBS (utilizing a huge β)
- IMSBS-GREEDY: a variant of IMSBS with a fixed beam-width $\beta = 1$ for the forward BS, performing a larger number of beam search (impact of iterations on the overall performance of IMSBS)
- IMSBS: a tuned version; configured to use an average runtime comparable to that of BS

Benchmark set RANDOM

For each combination of instance parameters

- $n \in \{50, 100, 200, 500\}$
- $m \in \{2, 3, 5, 10\}$
- $|\Sigma| \in \{2, 4\}$

10 random problem instances are generated (sequences uniformly at random).

The gap constraints generated from $G_s(\cdot) \in \mathcal{U}(\{\lfloor 0.5 \cdot |\Sigma| \rfloor, \ldots, \lfloor 1.5 \cdot |\Sigma| \rfloor\})$.

$\Longrightarrow$ A total of **320 problem instances** is generated.

Parameter tuning of IMSBS

We fixed (less sensible) params:

- $\textsc{Bs}$ (backward): $\beta' = 10$, and UB_2 (efficient)
- Candidate root nodes from $\mathcal{R}$ ordered by UB_2 (decreasingly)
- At each iteration, **10 best nodes** taken from $\mathcal{R}$ as the initial beam

Tuned parameters:

- β (in BS-forward)
- Heuristic guidance in BS (forward)
  - $\{\mathrm{UB}_1, \mathrm{UB}_2, h_{prob}\}$

BS: influence of different β and heuristic guidances

TODO: maybe to remove the plot

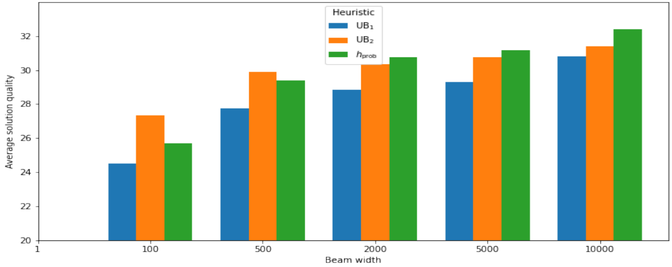
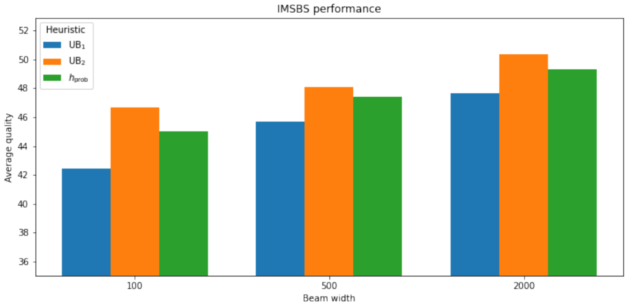


Figure: Avg. quality over all instances from the RANDOM benchmark suite.

(Baseline) BS $\Longrightarrow \beta = 10,000$ and $h = h_{\text{prob}}$

Parameter tuning of IMSBS



Avg. quality for different IMSBS settings over all instances from the RANDOM benchmark suite.

IMSBS $\Longrightarrow$ $h = \mathrm{UB}_2$ and $\beta = 500$, and *beam_iter* $= 100$.

IMSBS-GREEDY: $\beta = 1$, *beam_iter* $= 10,000$ (comparable/slightly larger avg. runtime to that of IMSBS)

Numerical results

Inst.			Bs		Imsbs-Greedy		Imsbs	
m	n	$\|\Sigma\|$	$\overline{obj}$	$\overline{t}[s]$	$\overline{obj}$	$\overline{t}[s]$	$\overline{obj}$	$\overline{t}[s]$
2	50	2	33.6	0.02	33.1	0.00	**37.7**	0.06
2	50	4	30.1	0.98	27.7	0.00	**30.1**	0.16
2	100	2	48.9	2.07	64.5	0.01	**72.8**	0.94
2	100	4	**62.1**	11.19	56.9	0.01	61.6	0.91
2	200	2	99.1	18.56	95.5	0.02	**136.4**	6.21
2	200	4	120.5	38.15	116.1	0.05	**124.9**	6.58
2	500	2	65.3	23.27	153.6	0.07	**265.7**	119.75
2	500	4	214.6	163.69	227.7	0.12	**294.4**	60.49
3	50	2	17.5	0.03	27.2	0.00	**31.2**	0.14
3	50	4	21.7	0.18	21.5	0.00	**22.9**	0.19
3	100	2	19.7	0.06	41.8	0.01	**58.5**	3.15
3	100	4	34.1	2.35	43.4	0.03	**48.4**	5.58
3	200	2	15.2	0.16	63.6	0.02	**90.0**	22.48
3	200	4	85.3	23.45	77.2	0.08	**97.1**	72.25
3	500	2	12.6	0.10	69.9	0.07	**102.9**	53.56
3	500	4	90.7	86.35	104.2	0.29	**187.7**	412.27
5	50	2	4.8	0.00	14.9	0.00	**20.0**	0.36
5	50	4	8.9	0.01	13.6	0.06	**15.3**	0.79
5	100	2	6.3	0.01	17.7	0.01	**22.4**	0.57
5	100	4	5.3	0.01	**23.2**	10.85	22.1	1.44
5	200	2	5.3	0.01	21.6	0.03	**26.6**	1.07
5	200	4	6.4	0.02	**32.5**	604.11	25.7	2.10
5	500	2	5.9	0.10	25.5	0.14	**27.9**	3.22
5	500	4	6.8	0.10	**43.6**	1341.25	26.9	3.52
10	50	2	1.7	0.00	8.8	2.28	**9.1**	0.47
10	50	4	1.9	0.00	**7.0**	508.36	6.1	1.46
10	100	2	1.1	0.00	**14.0**	1421.10	8.6	0.54
10	100	4	2.2	0.01	**8.9**	1800.45	6.3	1.51
10	200	2	2.5	0.01	**13.2**	1710.49	10.3	0.77
10	200	4	2.2	0.02	**7.9**	1800.54	6.1	1.74
10	500	2	1.8	0.08	**13.8**	1611.70	9.5	1.53
10	500	4	1.9	0.09	**8.2**	1800.46	6.1	2.37
Avg.			32.38	10.97	46.82	394.14	**59.73**	24.63

Numerical results for $m = 2$ case

- DP-1: the basic dynamic programming ($O(n^2m^2)$)
- DP-2: an advanced dynamic programming, uses Incremental Suffix Maximum Queries (ISMQ) with `Col` and `All` matrices for acceleration ($O(n^2 + mn)$)
- DP-3: an enhanced dynamic programming that handles ISMQ with a *dequeue* data structure (slightly modified w.r.t. the literature)
- ILP: an integer linear programming, **proposed in this work**, motivated by the ILP model for LCSP with $m = 2$

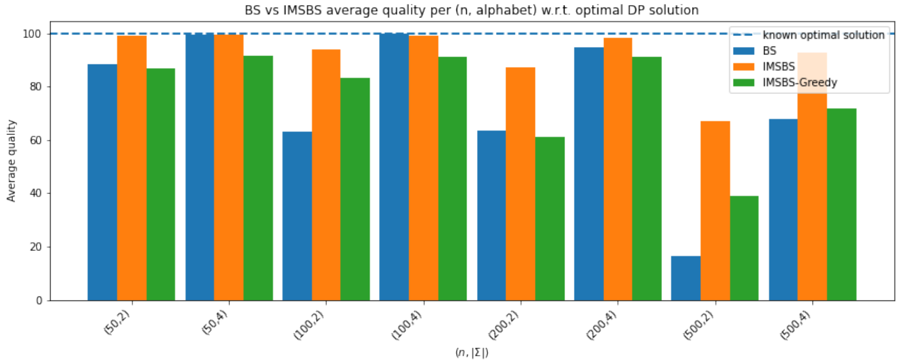
The first known empirical comparison for the $m = 2$ (80 instances).

Experimental evaluation

Table: Results on the RANDOM benchmark set for $m = 2$: the exact approaches from the literature.

Inst.			DP-1		DP-2		DP-3		ILP	
m	n	$\|\Sigma\|$	$\overline{obj}$	$\overline{t}[s]$	$\overline{obj}$	$\overline{t}[s]$	$\overline{obj}$	$\overline{t}[s]$	$\overline{obj}$	$\overline{t}[s]$
2	50	2	38.1	0.01	38.1	**0.01**	38.1	0.01	38.1	168.3
2	50	4	30.3	**0.01**	30.3	0.02	30.3	0.02	30.3	28.0
2	100	2	77.4	0.1	77.4	**0.03**	77.4	0.05	–	–
2	100	4	62.3	0.07	62.3	**0.06**	62.3	0.09	0.00	1800.0
2	200	2	156.4	0.75	156.4	**0.13**	156.4	0.16	–	–
2	200	4	127.2	0.59	127.2	**0.25**	127.2	0.32	–	–
2	500	2	395.9	13.57	395.9	**0.84**	395.9	1.05	–	–
2	500	4	317.2	10.18	317.2	**1.70**	317.2	2.1	–	–

The $m = 2$ case: heuristic performance vs. optimal solution



Relative solution quality achieved by the heuristic approaches compared to the optimal solutions.

Conclusions and Future Work

- Proposed a **general heuristic framework** IMSBS to solve the multiple VGLCS problem
- Combines Beam search calls (backward-and-forward manner) in an **iterative way** while producing promising source nodes for further BS iterations
    - Balancing intensification and diversification
- Empirical studies conducted for the first time on the **synthetic instances**: IMSBS wins over the baseline Beam search

Future work:

- **Real-world** instance-case scenario
- Lack of more advanced heuristic guidance: **Data-driven/ML heuristic** involving various local and global features (NN-based)

Thank you for your attention!