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, Bosnia and Herzegovina
[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 & Preliminaries
- Problem Definition
- Graph state space
- Iterative multi-source Beam Search
- Experimental Evaluation
  - General problem
  - Special problem
- Conclusions

Introduction

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

Longest common subsequence problem (LCSP)

- Basic problem in Computational biology
- Intensively solved over last 50 years
    - theoretically as well as practically
    - Practically: many approximation algorothms, (meta-) heuristics, exact approaches, etc.

Definition (LCSP)

Input: Given a set of sequences $S = \{s_1, \ldots, s_m\}$
Task: Find a subsequence s which is common for all sequences from S of maximum possible length.

Example

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

Literature & LCS Problem Variants

Theoretically:

- When $m = 2$ – polynomially solvable (in $O(n^2)$): Dynamic programming, Hunt-Schlimanski, ...
- 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,...)

Problem-related variants:

- Arc-annotated LCS problem
- Constrained (restricted/imposed) LCS problem
- Repetition-free, Longest filled LCS,...
- **Gapped LCS problem**

The gaped LCS problem

Definition (A gap sequence)

Given is a sequence s and an assigned function $G_s\colon \{1,\ldots,|s|\} \mapsto \mathbb{N}$. An ordered 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 gapped 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

$s = \texttt{AATTGC},\ G_s(\cdot) = 1$

- $\tilde{s} = \texttt{ATG}$, the embedding: AATTGC (valid gapped subsequence)
- $\tilde{s} = \texttt{ATG}$, 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} for each $i = 1, \ldots, m$

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

Literature & Motivation

- Peng and Yang (2012, 2014): studied the problem with $m = 2$ (poly-version): **Three dynamic programming** approaches (basic and sparse, sparse + advanced data-structures)
- Manea et al. (2024): Complexity bounds, (parameterised) complexity analysis investigated
- NP-hard under arbitrary m

Motivation:

- **Genetics and molecular biology**: applications in DNA and 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 the invalid extensions (edges) among the LCS extensions immediately
- Reach-ability of all extensions is an issue (**many root nodes** in the state graph, possibly separated subgraphs)

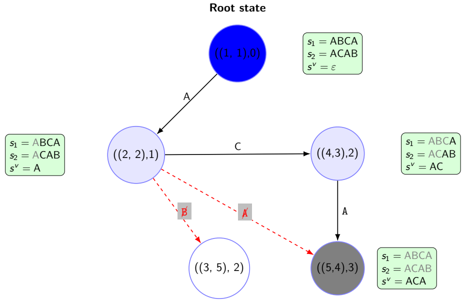
Root-based state graph formulation: rough idea

- Each **state** $v = (p^L, l^v)$: one or more feasible partial solutions characterized by a vector of positions p^L referring to relevant suffixes of input sequences to further expand these sols, and by the current subsequence length l^v
- **Node expansion** of v: extend partial solutions feasibly by one letter (concatenation) in all possible ways (like in LCS case), respecting gap constraints
- **Non-expandable nodes**: complete solutions

Crucial: selection of position r for root node (corr. to empty part. sol.

$\Rightarrow$ possibly **(exponentially) many root nodes**.
State graphs space induced by root node $(r, 0)$ we denote by *Space(r)*.

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



Root state
((1, 1),0)
$s_1 = \text{ABCA}$ $s_2 = \text{ACAB}$ $s^v = \varepsilon$
A
((2, 2),1)
$s_1 = \text{ABCA}$ $s_2 = \text{ACAB}$ $s^v = \text{A}$
C
((4,3),2)
$s_1 = \text{ABCA}$ $s_2 = \text{ACAB}$ $s^v = \text{AC}$
B
A
A
((3, 5), 2)
((5,4),3)
$s_1 = \text{ABCA}$ $s_2 = \text{ACAB}$ $s^v = \text{ACA}$

Figure: State space graph $Space(r = ((1, 1), 0))$ for MVGLCSP between the sequences ABCA and ACAB, assuming $G_1 = G_2 = 1$.

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

- The VGLCSP may exhibit multiple, potentially exponentially many, *disconnected root components* in its complete state space.
- If started from $(1, 1)$, it fails to reach optimal solutions.

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.

Iterative multi-source beam search (IMSBS) strategy

- **Full state space** of a VGLCSP instance: $\bigcup_{v\in\mathcal{R}} \mathrm{Space}(\mathbf{p}^{L,v})$, with $\mathcal{R}$ all root states
- Explicitly enumerating all root states computationally prohibitive (requiring $O(n^m)$ time)
- **IMSBS** (idea):
  - Based on **beam search**: limited breadth-first-search (BFS) strategy (for exploring *Space(r)*), parameter β controls the number of nodes to be further pursued
  - dynamically explore multiple promising regions of the state space: move from one rooted subgraph to another one
  - **iteratively identifies and expands** a set of promising candidate root states

Components and characteristics of IMSBS

- $\mathcal{R}$: a global set of root-node candidates, dynamically updated over iterations
- **Beam search** (backward): not all candidate nodes from $\mathcal{R}$ are root nodes
  - **refine** them by performing the **backward** approximate passes from candidate-root nodes to effectively reach distant (real) and promising root node in the state space
- **Beam search** (forward): intensification in the chosen root-space *Space(p)*: seek for high-quality (complete) solutions in the corresponding search sub-regions
- diversification: during each BS (forward), the complete nodes are expanded in all possible ways (gaps ignored) to generate candidates for roots (distant from the current region)

Working example

TODO: draw some plots regarding the workflow of the approach

The core advantages of the IMSBS approach

- Balance between complete beam search execution (local exploration of promising paths) and the generation of new source nodes (**global coverage** of the search space)
- Preventing from direct suboptimal solutions reduced by executing backward beam search procedure (**refinement**)

Heuristic guidances in BS

Three LCS heuristic guidances used:

- *"Look-ahead" for the remaining length:*

$$\mathrm{UB}_1(\mathbf{v}) = l^v + \min_{i=1,\dots,m} (|s_i| - p_i^v + 1) \tag{1}$$

- *Character Frequency Alignment:*

$$\mathrm{UB}_2(\mathbf{v}) = l^v + \sum_{\sigma\in\Sigma} \min \underbrace{\left(|s_1[p_1^v, |s_1|]|_\sigma, \dots, |s_m[p_m^v, |s_m|]|_\sigma\right)}_{\text{remaining valid suffixes}} \tag{2}$$

- *A probability-based heuristic* guidance h_{prob}: by the pre-processed matrices of probabilities (Mousavi and Tababa (2012)). These probabilities approximate the probability for the event that a sequence s of length k is a subsequence of a (random) sequence of length n

Experimental Evaluation: arbitrary large *m*-case

- Bs: a baseline beam search approach, allowing **a single iteration** of IMSBS (utilizing a huge β)
- Imsbs-greedy: a variant of IMSBS with a fixed **beam-width parameter** $\beta = 1$ for the forward BS. This configuration highlights the impact of performing a larger number of beam search 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 such that $G_{s_i}(j) \in \mathcal{U}(\{\lfloor 0.5 \cdot |\Sigma| \rfloor, \ldots, \lfloor 1.5 \cdot |\Sigma| \rfloor\})$.

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

Parameter tuning of IMSBS

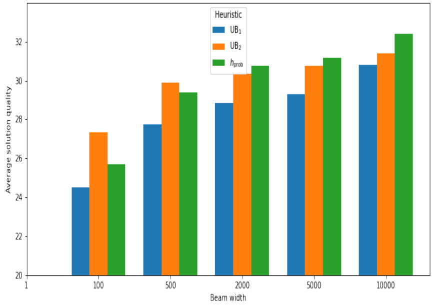
We fixed:

- $\textsc{Bs}$: $\beta = 10$, and UB for the backwards BS (efficient)
- Candidate root nodes from $\mathcal{R}$ ordered by *UB* (decreasingly)
- At each iteration, 10 best nodes from $\mathcal{R}$ taken

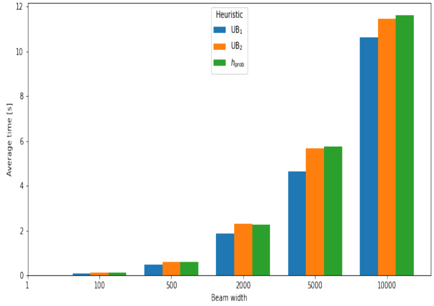
Tuned parameters:

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

Bs: influence of β and heuristic guidances



(a) Avg. quality over all instances from the Random benchmark suite.

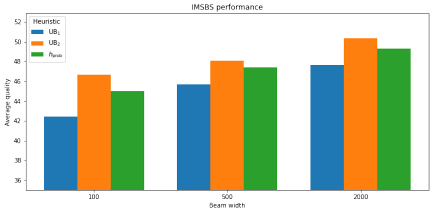


(b) Avg. runtime over all instances from the Random benchmark suite.

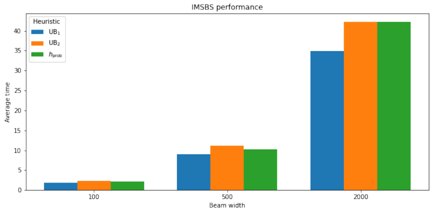
Figure: Beam search tuning results on the Random benchmark suite.

$\Longrightarrow \beta = 10,000$ and $h = h_{\text{prob}}$

Parameter tuning of IMSBS



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



(b) Avg. runtime for different IMSBS settings over all instances from the RANDOM benchmark suite.

Figure: IMSBS parameter tuning results on the RANDOM benchmark suite.

$\Longrightarrow h = \mathrm{UB}_2$ and $\beta = 500$.

IMSBS-GREEDY: $\beta = 1$, *imsbs_iter* $= 10,000$ (comparable avg. runtime to IMSBS)

Numerical results

content...