# 1 LCS-Type Problems for Sets of Biological Sequences

In all definitions below, let

$$\mathcal{S} = \{s_1, s_2, \ldots, s_m\}, m \in \mathbb{N}$$

be a set of DNA, RNA, or protein molecules, where each $s_i$ represents a sequence over an alphabet $\Sigma = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}/\mathrm{U}, ...\}$. A sequence $s$ is a *subsequence* of $M_i$ (denoted $s \sqsubseteq s_i$) if the characters of $s$ appear in $s_i$ in the same order but not necessarily contiguously. E.g., given is $s_1$="ACCTAA", one its subsequence is given by "ACA". As follows is a list of optimization problems from bioinformatics that are involved in capturing structural similarities between arbitrary large set of molecular sequences $\mathcal{S}$.

## 1.1 Longest Common Subsequence (LCS) Problem

**Input:** A set of molecules $\mathcal{S} = \{s_1, \ldots, s_m\}$, the basic variant comes with $m = 2$.

**Objective:** Find the longest sequence $s$ such that

$$s \sqsubseteq s_i \quad \forall i.$$

This represents **the longest sequence** of nucleotides preserved across all molecules.

This problem generalizes LCS to more than two sequences ($m$) and is commonly used for multi-species motif analysis.

## 1.2 Constrained Longest Common Subsequence (CLCS) Problem

**Input:**

- A set of molecules $\mathcal{S}$.
- A set of required motifs $\mathcal{C}^+ = \{p_1, \ldots, p_r\}$.
- A set of forbidden motifs $\mathcal{C}^- = \{q_1, \ldots, q_s\}$.

**Objective:** Find the longest common subsequence $s$ such that:

- $s \sqsubseteq s_i \quad \forall i$
- $p_j \sqsubseteq s \quad \forall P_j \in \mathcal{C}^+$
- $q_\ell \not\sqsubseteq s \quad \forall q_\ell \in \mathcal{C}^-$

This **captures motif-inclusion or exclusion** constraints often used in biological analysis.

### 1.3 Restricted Longest Common Subsequence (RLCS)

**Input:**

- A set of sequences $\mathcal{S}$.
- A set of forbidden match pairs

$$\mathcal{R} \subseteq \bigcup_{i<j} \big(\mathrm{pos}(s_i) \times \mathrm{pos}(s_j)\big),$$

  meaning certain positional matches are disallowed.

**Objective:** Find the longest sequence $S$ common to all molecules such that none of the matched positions between any pair $(s_i, s_j)$ belongs to $\mathcal{R}$. RLCS models **biologically constrained situations** where **certain nucleotides cannot be aligned**.

### 1.4 Gap-Constrained LCS (GC-LCS)

**Input:**

- Sequences $\mathcal{S}$.
- A gap value $g \in \mathbb{N}$ receiving an integer gap value from the interval $[L, U] \cap \mathbb{N}$;

**Objective:** Find a common subsequence $s$ such that if $s_t$ and $s_{t+1}$ match positions $a_i$ and $b_i$ (the nucleotide at these positions fit) in a molecule $s_i$, then the gap between them satisfies

$$L \leq (a_i - b_i - 1) \leq U, (\forall i = 1, \ldots, |s|)$$

Gap constraints enforce **biologically plausible spacing** between matched nucleotides.

### 1.5 Variable-Gap LCS (VGLCS)

This is a generalized variant of the problem above.

**Input:**

- Sequences $\mathcal{S}$.
- A gap function $g_i$ for each sequence $s_i \in \mathcal{S}$, is given as the mapping

$$g_i \colon \{1, \ldots, |s_i|\} \mapsto [L, U] \cap \mathbb{N}$$

  receiving an integer gap value from the interval, i.e., $g_i(\cdot) \in [L, U]$ where $L, U \in \mathbb{N}$;

**Objective:** Find a common subsequence $s$ that respects **all gap constraints**. That is, if the nucleotides of $s$ appear on positions $p_1^i <\ldots,< p_{|s|}^i$ in $s_i$ then $|p_j^i - p_{j-1}^i| \leq g_i(p_j^i) + 1$, for each $j = 2, \ldots, |s|$, and each $i = 1, \ldots, m$. Shortly, for each two consecutive nucleotides of $s$ and their respective positions $(p_1^i, p_2^i)$ in $s_i$, the longest possible distance between them is given by the number $g_i(p_2^i)$. In other words: the distance between two consecutive symbols of the subsequence has to be between bounded above by an upper bound depend on the position of those symbols in the subsequence.

This is suitable for motifs where each structural region has different spatial constraints (e.g., **RNA secondary structure**).

### 1.6 Examples Using DNA/RNA Sequences

Consider the set:

$$\mathcal{S} = \{s_1 = \text{ACGTA},\ s_2 = \text{AGTCTA},\ s_3 = \text{ACTAG}\}.$$

*LCS:* One longest common subsequence for two molecules is:

$$S = \text{ACTA}.$$

*CLCS (must include $C^+ = \{\text{G}\}$):* The result would be s="AG"

*RLCS (forbid matching C in $s_1$ with C in $s_2$):* A valid restricted LCS is:

$$s = \text{ATA}.$$

*Gap-Constrained LCS (max gap = 1):* A possible GC-LCS is:

$$s = \text{ATA}.$$

*MLCS:* For all three molecules, the result is

$$s = \text{ACTA}.$$

## 2 SOTA methodology for solving LCS-based problems & Further Ideas

### 2.1 The basic (or multiple) LCS problem

This problem is the most basic, and most known in this area of research. It is a constraint-free and solve by dozens of approaches in the literature from approximation algorithms, exact approaches, and several anytime hybrid variants, and pure metaheuristic approaches, tree-based or trajectory-based.

Among all of these, Beam search framework has shown as the most robust approach. Based on the general state-graph framework, whihc depends only by two parameters, the integrated (problem-specific) heuristic to guide the overall search process, there is just one parameter to be tuned — number of nodes to be further pursued for expansion and generation of nodes for the subsequent level. Beam search is a tree-based meta-heuristic approach that performs in a breadth-first-search manner. It means, it always expands up to a number (parameterized by $\beta$) most promising nodes according to some criteria of evaluation (heuristic evaluations). The most efficient heuristic so-far are: ($i$) the expected-length approximation heuristic; ($ii$) the GMPSUM heuristic that includes two scores as a convex combination ... Furthermore, to make the Beam search approach even more robust, a dynamic version of it has been proposed by dynamically adjusting beam-width ($\beta$) per each level depending on the remaining time estimation and the current beam size. Concerning the exact solving techniques, an anytime hybrid of A$^*$ and Anytime column search (ACS) has a best performing profile.

### 2.2 Constrained LCS Problem

For the constrained version of the LCS problem, Beam search again has shown the best performance. The generalized search framework has been first proposed, along with adapted heuristic guidances specifically designed for this kind of problem, utilizing also constraints and their influence in the final evaluation score. An extension of the expected-length approximation heuristic has proved as the best in terms of the final solution quality. For the fixed-problem with two input strings and one inclusion patters, A$^*$ search methodology has been most efficient.

### 2.3 Variable Gapped LCS Problem

This interesting variant of the problem has been solved for the fixed version with two input strings by means of three different versions of dynamic programming: the basic one, the sparse variant, and more advanced variant employing complex data structure for speeding up the overall calculation. Apart from that, we are not aware of any approach tackling the version with arbitrary large set $\mathcal{S}$. We are going to design the approach in our EUROCAST paper that proposes so-called *the Multi Source Beam search approach* that operates on a set of generally weakly connected state-space graph with multiple source nodes.

### 2.4 Gap-constrained LCS Problem

The polynomial version with two sequences is solved by dynamic programming in $O(n^3)$ time, where $n$ denotes the length of longer sequence, too slow for long sequences. It belongs to the sequence alignment (SA) as one of the most fundamental components of bioinformatics. This represents a relaxed version of SA. More advanced DP is given by incorporating the Segment Trees Optimization. Graph-Based formulation is yet another popular modern interpretation that consists of build a DAG of match pairs, add edges only when the gap constraint is satisfied. Afterwards, solving longest path in DAG, which is linear in edges. The methodology is also used in Protein sequence analysis. Heuristics and Meta-heuristics are plan to to used (TODO: planning to do). For very large sequences or for multi-constraint variants: Greedy heuristics and Beam search is worth of a try. More advanced ideas – LLM-Assisted Heuristics: Using LLMs to score promising extension paths in the match-pair graph; additionally, ML models to predict pruning thresholds.

## 3 Enzyme study and Optimization/ML: rough ideas

Suggestions are made w.r.t. provided dataset of Enzymes (`.PDB` format) and mainly relate do the combination of Optimization techniques and Machine Learning in studying enzymes, their structure, understanding functionality, evolution, etc. A few rough/pre-mature ideas are exposed here.

**Structural Alignment & Clustering**: Instead of sequential structures and several optimization problems that identify structurally similar motifs between a bunch of molecular structures, here we deal with more dynamic structures (in essence, graphs). Therefore, one will use structures instead of sequences in comparisons. By keeping this in mind, what could be done is:

– RMSD-based clustering
– Fold classification
– Discovery of novel enzyme families

**Enzyme mimic**:

– This strongly suggests: Designed proteins, enzyme mimics
– Possibly AlphaFold / Rosetta outputs
– Design Validation: Compare designed enzymes vs natural enzymes (Are there any automatic ways of doing so?)
  **Measure indicators:** Pocket volume, Catalytic residue geometry, **Stability indicators** (Domination-based optimization problems)

Research question: Do designed enzyme mimics reproduce native catalytic geometry?

**Machine Learning / AI on 3D Enzyme Data**:

– Convert each enzyme into a graph:
  - *Nodes* – residues or atoms
  - *Edges* – spatial proximity
– Train models (GNNs (PyTorch Geometric), 3D equivariant networks) to:
  - Predict enzyme class
  - Predict activity
  - Predict stability – a lot of annotated data required, do we have them?

**Structure → Function Prediction**:
Train ML models using: 3D features, Pocket descriptors, Electrostatic surfaces (extraction of features?)
Target labels: EC number, Substrate class, Catalytic mechanism, etc.
**Evolutionary & Comparative Studies**:

- **Structure-Guided Evolution**
  - Map sequence variation onto structure
  - Detect structurally conserved "hotspots"
- Design vs Evolution: Compare natural enzymes to designed ones, Quantify "distance from natural fold space"

Best Research Direction for myself (based on my profile): Metaheuristics, Beam search, ML-guided search, ...
More precisely, the idea could be: **Guided exploration of enzyme design space using structure-aware heuristics or ML-guided search.**

**Another idea:** structural analysis of emzyme based on graph-based instance features (domination problems, stability, diameters, ... Learn stability/function predictors).

## 3.1 Conversion into Graphs

**A protein/enzyme graph** is typically:

- Nodes → atoms or residues
- Edges → spatial proximity, chemical bonds, or interactions
- Node features → residue type, atom type, coordinates, etc.
- Edge features → distance, bond type, interaction strength

There is no single graph definition — the choice depends on your research goal.
**Residue-level graph** (most popular): Best for ML, GNNs, classification, design

- Nodes are amino acid residues
- Edges are residues within distance threshold (e.g. $\leq 8$)
- Node features
  - One-hot AA type
  - Secondary structure
  - Hydrophobicity

Edge features:

- Euclidean distance, Contact type

Residue-level graphs are used in AlphaFold-like and GNN models.

**Atom-level graph** (high resolution): Best for chemistry, docking, catalysis

- Node = atom
- Edge = covalent bonds + spatial contacts
- Node features
  - Element, Partial charge, Atom type
- Edge features
  - Bond order, Distance

Atom graphs are much larger graphs, inquiring for higher computational cost.

**Pocket / active-site graph**: Best for enzyme function prediction

- Nodes = residues in pocket
- Edges = distances / interactions
- Focused, interpretable

With enzyme graphs we can do:

- Predict enzyme class (EC number)
- Compare natural vs designed enzymes
- Detect catalytic motifs
- **Learn stability/function predictors**
- Guide enzyme design via ML

This aligns perfectly with:

- search / heuristics
- ML-guided exploration
- structure-aware optimization