Longest Common Subsequence with Gap Constraints (LCS-FIG)

Mohammad Sadegh Sirjani — zxl708

March 4, 2025

Abstract

This document defines the Longest Common Subsequence with Gap Constraints (LCS-FIG) problem and presents three distinct algorithms for solving it. The problem requires finding the longest subsequence common to two sequences such that the gap between consecutive matching characters does not exceed a fixed parameter K. Three approaches are described: (1) a dynamic programming solution (FIG-DP), (2) an optimized DP solution using range maximum queries (RMQ-FIG), and (3) a greedy approximation. Each algorithm is detailed with pseudocode and complexity analysis.

1 Problem Definition

Given two sequences X and Y, the goal is to find the longest common subsequence Z subject to the constraint that for any two consecutive elements of Z, if $Z[i-1]$ appears at position p in X (or q in Y), then $Z[i]$ must appear no later than position $p+K+1$ in X (and similarly in Y), where K is a fixed gap parameter. This variant is especially useful in bioinformatics and text analysis where controlled spacing between matches is required.

2 Algorithm

2.1 FIG-DP (Dynamic Programming)

This algorithm modifies the classic DP approach for LCS by adding a gap constraint. When characters match, it checks if the previous match is within the allowed gap (K). It updates the LCS length accordingly, ensuring valid subsequences.

Algorithm 1 FIG-DP Algorithm for LCS-FIG

```
Input  : Sequences $X[1 \dots n]$, $Y[1 \dots m]$, fixed gap $K$
Output: Length of LCS-FIG
Initialize $T[0 \dots n][0 \dots m] \leftarrow 0$  for $i \leftarrow 1$ $n$ do
    for $j \leftarrow 1$ $m$ do
        if $X[i] = Y[j]$ then
            if $i > K + 1$ and $j > K + 1$ then
                $T[i][j] \leftarrow T[i - K - 1][j - K - 1] + 1$
            else
                $T[i][j] \leftarrow 1$
        else
            $T[i][j] \leftarrow \max(T[i-1][j], T[i][j-1])$
return $\max(T[i][j] : 1 \leq i \leq n, 1 \leq j \leq m)$
```

Time Complexity: $O(nm)$
Space Complexity: $O(nm)$
Reference: [2]

2.2 RMQ-FIG (Optimized DP with Range Maximum Query)

RMQ-FIG enhances the DP approach by using a range maximum query (RMQ) structure to quickly find the best LCS length within the valid gap range. This optimization speeds up the algorithm compared to standard DP.

Algorithm 2 RMQ-FIG Algorithm for LCS-FIG

```
Input  : Sequences $X[1 \dots n]$, $Y[1 \dots m]$, fixed gap $K$
Output: Length of LCS-FIG
Initialize 2D table $T[0 \dots n][0 \dots m] \leftarrow 0$ and RMQ structure  for $i \leftarrow 1$ $n$
 do
    for $j \leftarrow 1$ $m$ do
        if $X[i] = Y[j]$ then
            $prev\_best \leftarrow$ RMQ.query($\max(0, i-K-1), i-1, \max(0, j-K-$
              $1), j-1)$  if $prev\_best$ is valid then
                $T[i][j] \leftarrow prev\_best + 1$
            else
                $T[i][j] \leftarrow 1$
            RMQ.update($i, j, T[i][j]$)
        else
            $T[i][j] \leftarrow \max(T[i-1][j], T[i][j-1])$
return $\max(T[i][j] : 1 \leq i \leq n, 1 \leq j \leq m)$
```

Time Complexity: $O(nm \log n)$
Space Complexity: $O(nm)$
Reference: [3]

2.3 Greedy Approximation for LCS-FIG

This greedy algorithm approximates the LCS-FIG solution by selecting matching characters and jumping $K + 1$ positions ahead in both sequences. It provides a fast but non-optimal solution.

Algorithm 3 Greedy LCS-FIG Algorithm

```
Input  : Sequences $X[1 \ldots n]$, $Y[1 \ldots m]$, fixed gap $K$
Output: Length of LCS-FIG
Initialize $i \leftarrow 1$, $j \leftarrow 1$, $LCS\_length \leftarrow 0$ while $i \leq n$ and $j \leq m$ do
    if $X[i] = Y[j]$ then
        $LCS\_length \leftarrow LCS\_length + 1$ $i \leftarrow i + K + 1$ $j \leftarrow j + K + 1$
    else if $X[i] < Y[j]$ then
        $i \leftarrow i + 1$
    else
        $j \leftarrow j + 1$
return $LCS\_length$
```

Time Complexity: $O(n + m)$
Space Complexity: $O(1)$
Reference: [1]

3 Conclusion

Three distinct algorithms for the LCS-FIG problem have been presented. The FIG-DP algorithm provides an exact solution using a modified DP table. The RMQ-FIG algorithm optimizes the DP approach with range maximum queries, and the Greedy Approximation offers a fast, approximate solution. Each method has its own trade-offs in terms of time and space complexity.

References

[1] Alireza Abdi and Mohsen Hooshmand. Longest common subsequence: Tabular vs. closed-form equation computation of subsequence probability. *arXiv preprint arXiv:2206.11726*, 2022.

[2] Duncan Adamson, Maria Kosche, Tore Koß, Florin Manea, and Stefan Siemer. Longest common subsequence with gap constraints. *arXiv preprint arXiv:2304.05270*, 2023.

[3] Radu Stefan Mincu and Alexandru Popa. Heuristic algorithms for the longest filled common subsequence problem. *arXiv preprint arXiv:1904.07902*, 2019.