Giovanni Buzzega
Department of Computer Science
University of Pisa
Pisa, Italy

Indexing of Maximal Common Subsequences and their application

Definition

Maximal Common Subsequences (MCSs) are common subsequences (CSs) that are not subsequence of any other common subsequence.

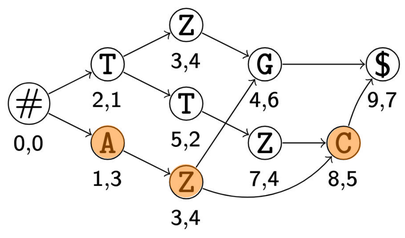
$$Z \in \mathrm{MCS}(X,Y) \Leftrightarrow \forall W \in \mathrm{CS}(X,Y), Z \not\subset W$$

X: #**A**T**Z**GTCZ**C**$
Y: #TT**AZC**G$

We can substitute (X,Y) with any collection of k strings.

Goal

Can we build a compact **deterministic index for all MCSs?**

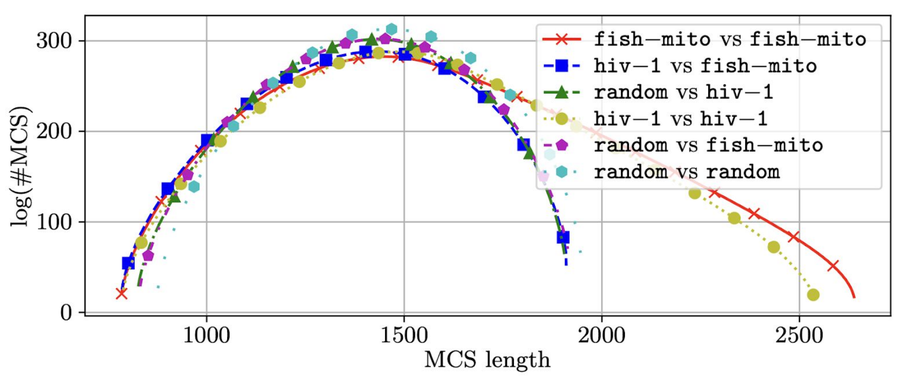


↪ Node-labeled Direct Acyclic Graph
↪ At most 1 out-neighbor per symbol
↪ Able to reconstruct MCS positions
↪ Efficient to construct in practice
↪ With query support

Why?

Why MCSs?

The length distribution of MCSs seems to correlate well with **sequence relatedness**:



It can be used to define a **distance on strings** and, when extended to k strings, it can provide a new **string clustering** method.

All **Longest Common Subsequence** (LCS) are also maximal, as by definition they cannot be extended with any character.

Why indexing?

Such an index allows to efficiently **retrieve any interesting MCS** without resorting to full enumeration.

Finding the **longest path is linear** in a DAG, so we can retrieve **LCSs**.

The associated position of each symbol could be used as the base of a Multiple Sequence Alignment.

There are many other MCSs that are **not of maximum length** that can provide equally good **alignment information**.

Complexity gains?:

Finding one **LCS** over k strings is **NP-hard**. (Mayer, 1978)[1]

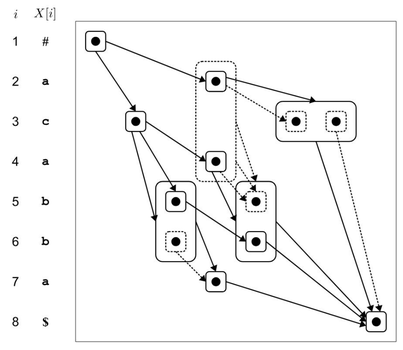
No **LCS** over **2** strings can be found in **strongly subquadratic** time.[2,3]

Generating one **MCS** over k strings is **efficient**:

O(kn log n) (Hirota and Sakai, 2023)[4]

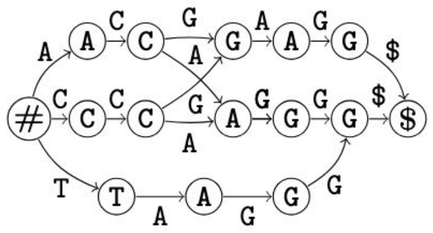
Can we use a *long-enough* MCSs to approximate the useful features of an LCS over k strings?

Previous solutions



(Hirota and Sakai, arXiv 2023):[5]
↪ Non-deterministic
↪ $O(n^3)$ time and space

(Conte *et al.*, 2023):[6]
↪ deterministic
↪ $O(n^3|\Sigma|\log n)$ time
↪ $O(n^3|\Sigma|)$ space



MCSs can be exponential, even for the case of two strings.
e.g.: X: A GGA GGA GGA… Y: A GA GA GA GA…

These are **polynomial-sized indices,** which are crucial for efficient storage and retrieval.
But:
Both solutions are built for **2 strings**
Both strive to build the index in one go, which makes the algorithmic choices hard to grasp

Our simplified approach: McDag[7]

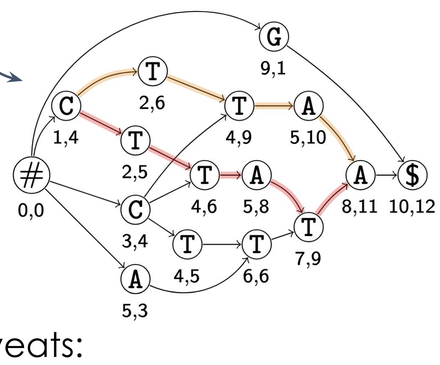
1) Build an *approximate* ***rightmost co-deterministic*** *MCS index*
   ↪ containing **all MCSs** and **some non-maximal CSs**
2) **Filter-out** all paths that correspond to **non-maximal CSs**

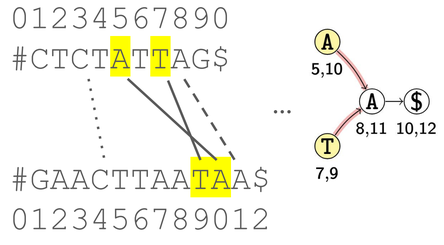
Generalizing to k strings is easy: each *match* is a k-ple of positions

How?

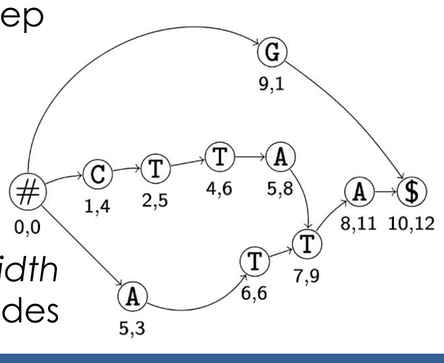
Start from the end of the strings
For each ***match*** find the rightmost occurrences of each symbol and add the corresponding node as in-neighbor
↪ at most 1 in-neighbor per symbol
↪ **co-deterministic**

In the resulting index, each non-maximal CSs has one MCS that contains it in a **subsequence-bubble**.



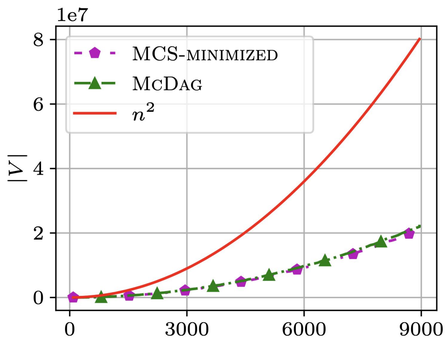


We can remove all subsequence-bubbles via a **determinization & filtering** step

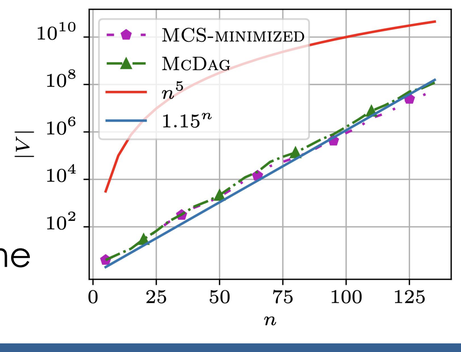


Caveats:
Determinization can be exponential unless the starting automaton has low *width*
One match can correspond to multiple nodes

Performances



Always 4-7% larger than the minimum-size deterministic MCS index, for 2 strings



For k>4 strings the trend seems to become exponential even for the minimal index

Open questions

Can we extend the concept of MCS over automata languages?
Can we build an approximation of the histogram of MCS lengths?

References

[1] Maier, D. The complexity of some problems on subsequences and supersequences. Journal of the ACM (JACM) 25, 2 (1978), 322–336.
[2] Abboud, A., Backurs, A., and Williams, V. V. Tight hardness results for lcs and other sequence similarity measures. In 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (2015), IEEE, pp. 59–78.
[3] Bringmann, K., and Künnemann, M. Quadratic conditional lower bounds for string problems and dynamic time warping. In 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (2015), IEEE, pp. 79–97.
[4] Hirota, M., and Sakai, Y. A fast algorithm for finding a maximal common subsequence of multiple strings. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 106, 9 (2023), 1191–1194
[5] Hirota, M., and Sakai, Y. Efficient algorithms for enumerating maximal common subsequences of two strings. arXiv preprint arXiv:2307.10552 (2023).
[6] Conte, A., Grossi, R., Punzi, G., and Uno, T. A compact dag for storing and searching maximal common subsequences. In 34th International Symposium on Algorithms and Computation (2023).
[7] Buzzega, G., Conte, A., Grossi, R., and Punzi, G. Mcdag: Indexing maximal common subsequences in practice. In 24th International Workshop on Algorithms in Bioinformatics (WABI 2024) (2024), Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

PhD Spotlight - 11th of November 2024 - Pisa