String comparison by transposition networks

Peter Krusche

Algorithms and Complexity Research Group

Overview and basic definitions

Computing string (or sequence) alignments [1, 3] is a classical method of comparing strings and has applications in many areas of computing, such as signal processing, bioinformatics, and source code management. Semi-local string alignment is a recent generalisation of this method, in which the alignment of a given string and all substrings of another string are computed simultaneously at no additional asymptotic cost.

Let $x = x_1 x_2 \dots x_m$ and $y = y_1 y_2 \dots y_n$ be two strings over an alphabet Σ of size σ. We distinguish between consecutive *substrings* of a string x which can be obtained by removing zero or more characters from the beginning and/or the end of x, and *subsequences* which can be obtained by deleting zero or more characters in any position. The *longest common subsequence* (LCS) of two strings is the longest string that is a subsequence of both input strings, its length (the LLCS) is a measure for the similarity of the two strings. A common approach to this problem is to reduce it to finding longest paths in directed acyclic grid graphs (see Figure 1).

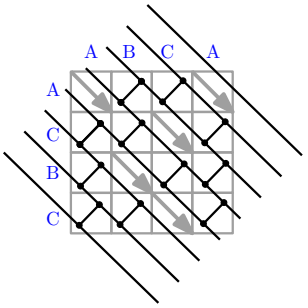

Fig. 1. Comparison network and alignment dag

For many applications, the LCS itself is of lesser interest than its length, particularly for different substrings, prefixes or suffixes of the input strings [4]. The result of such a computation is a *highest-score matrix* that contains LCS lengths of specific substrings of the input strings. Solutions to the semi-local LCS problem are given by a highest-score matrix A where $A(i, j)$ is defined as the LLCS of x and substring $y_i \dots y_j$.

In this talk, we expose a close connection between semi-local string comparison and a certain class of traditional comparison networks [2] known as transposition networks. Based on this, we will show how bit-parallel and parameterized algorithms for string comparison can be derived using this method, and how this relates to previous approaches.

References

1. Apostolico, A.: String editing and longest common subsequences. In: Handbook of Formal Languages. Volume 2. Springer-Verlag (1997) 361–398
2. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, Second Edition. The MIT Press and McGraw-Hill Book Company (2001)
3. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals,. Sov. Phys. Dokl. **6** (1966) 707–710
4. Tiskin, A.: Semi-local string comparison: Algorithmic techniques and applications. Mathematics in Computer Science, to appear. arXiv: 0707.3619