2D Alignment

Glencora Borradaile

The algorithm for sequence alignment is familiar to many undergraduates as an example of dynamic programming. Given two input sequences a and b over an alphabet Σ, an alignment is given by subsequences a' of a and b' of b such that $|a'| = |b'|$. Given distance cost $\delta : \Sigma^2 \to [0, 1]$ and skip cost σ, the goal is to find the alignment that minimizes the sum of the pairwise distances in a' and b' plus $\sigma(|a| - |a'| + |b| - |b'|)$. That is, find subsequences that are close to each other without having to remove too many elements.

We generalize this problem to the alignment of 2D arrays over the alphabet Σ, A and B. A subarray A' of A is achieved by deleting rows and columns of A. An alignment is given by subarrays A' of A and B' of B such that A' and B' have the same size. The goal is to find the alignment that minimizes the sum of

- σ times the number of rows and columns deleted from A to get A'
- σ times the number of rows and columns deleted from B to get B'
- the sum of the pairwise distances in A' and B'.

The only version of this problem that has been studied algorithmically is for the distance function $d(x, y) = \infty$ if $x \neq y$ and $d(x, x) = 0$ [1]. This problem is NP-hard and seems hard to approximate. That is, the best α-approximation[1] could be for α polynomial in the size of the input arrays, indeed a bad approximation. However, we might do better by taking into account the problems that 2D alignment can model:

Spreadsheet edit distance There is a spreadsheet C that we model as a 2D array. Two people, under version control, start editing this spreadsheet independently, resulting in spreadsheets A and B. The types of edits they do are: inserting a row, inserting a column, deleting a row, deleting a column, changing individual entries. We would like to align A and B (by highlighting the added columns/rows and the changed entries) to find the changes that have been made. The alignment of A and B should be a subarray of C (a subarray because rows and columns of C may have been deleted). Since A and B were generated by by exactly the edits our problem models, we expect there to be a particularly good optimal solution. Can we find it?

For such a problem in which the globally optimal solution is significantly better than any locally optimal solution, algorithms with poor worst-case approximation ratios perform very well in practice. This idea can be extended to an analysis using the *planted model*.

Simple image comparison You are given two images A and B and you want to determine if B was generated from A by cropping, scaling and small edits (ie. red-eye removal, adding text, other small photoshopping). Cropping is equivalent to deleting rows and columns at the edge of the image and scaling is (close to) deleting every k^{th} row/column (for several values of k).

In this application, the additional constraints on the types of rows and columns that can be deleted that may lead to better algorithms than would be possible for the general 2D alignment problem.

In this project, the student will help develop algorithms for these problem and implement the resulting algorithms. She will test these implementations on inputs derived from the above problems. The student will be involved in the theoretical analysis with myself and my collaborators on this problem.

References

[1] A. Amir, T. Hartman, O. Kapah, B. Shalom, and D. Tsur. Generalized LCS. In *Proc. of the 14th Symposium on String Processing and Information Retrieval*, pages 50–61, 2007.

[1] An α-approximation is an algorithm that finds, in polynomial time, a solution whose value is within α of optimal.