6.854 Final Project

Arsen Mamikonyan; Hayk Saribekyan

December 12, 2016

Abstract

In this paper we review and implement two algorithms presented by Bokal et al.[1], and Chan and Pratt [2], presented in SoCG'15 and SoCG'16 respectively. Both works introduce novel approaches to finding maximal subsequences with given hereditary properties.

1 Introduction

With increasing number of sensors and location-tracking devices, there are massive datasets describing movements of people, animals, robots, etc. Since the datasets mostly describe trajectories of movement, often each location point is associated with a timestamp. As a result, in the process of analysing such datasets, efficient algorithms for time range queries are required.

This paper discusses problems of the following nature: given a sequence of points $S = s_1, \ldots, s_n$, and a property P defined on a sequence of points. For any given range $[i, j]$, we want to answer whether P is true for $s_i, \ldots, s_j$. Additionally, we restrict P to be a hereditary property i.e. if P holds for sequence set A, then P also holds for any subsequence of A. The ability to answer these queries has practical applications and, in particular, it is the case for the two problems discussed in this paper.

Suppose we have a data structure that for any index i retrieves the largest integer $j^*(i)$ such that P is true for the sequence $S[i, j^*(i)]$. Then, it is trivial to check whether P holds for $S[i, j]$ as one only needs to compare j and $j^*(i)$. Therefore, the time range query problem reduces to building such data structure. Each of the two papers reviewed here present a general framework that can be used to build such data structures for a variety of problems.

[1, 2] use their frameworks to efficiently solve the range query problem for different properties P. We have implemented one algorithm from each paper. The first one is the *monotonicity* property. A sequence of points S in two-dimensional space is monotone if there is a line l, for which if points of S are projected on l, their order is maintained. The algorithm described in [1] is linear. The second one is the *clustering* property. We will call a set of points S clustered, if the largest distance between any pair of them (the diameter of S) is at most 1. [2] present a $O(n \log n)$ algorithm to solve for this property.

For both of the problems there is a simple algorithm to solve them in quadratic time. We discovered that, while the algorithms presented in [1, 2] are elegant and efficient, their implementation can be extremely complicated with many edge cases. Nevertheless, they are much faster for large enough datasets.

2 Our contribution

We've implemented

- In $O(n)$ time we can find all maximal subsequences that define monotone paths in some (subpath-dependent) direction. [1]
- In $O(n \log^2 n)$ time time we can find all maximal subsequences with diameter at most 1. [2]

3 Algorithms

3.1 k^*

Let $k^*(i) = \inf_{m \geq i}\{d(i,m) > 1\}$. **Claim** $j^*(i-1) = \min(j^*(i), k^*(i-1))$. Thus after we calculate $k^*(i)$ for all elements, we can calculate $j^*(i)$ in $O(n)$ time by looping over all indices in the reverse order.

3.2 Bokan et al Overview

Upper triangle method

3.3 Chan, Prat Overview

Range tree method.

4 Implementation Details

Talk about sweep line, etc.

5 Experimental Results

5.1 Monotonicity

Figure 1: Monotonicity algorithm results for moving point with Gaussian noise.

First let's look at the monotonicity resutls. Here we use the following dataset.

We have a point moving along x axis in the positive direction with speed 1 (i.e. 1 per index), we add Gaussian noise to this point to make the problem interseting. In Figure 1 you can see results for noise with standard deviations $\sigma_y = 0.05, \sigma_y = 1$. After we generate the dataset, we run Bokal et. al algorithm on the dataset and the answer boundarries of the matrix A that indicates if for points in range $[i, j]$ exists a common direction that has positive dot product with each of the displacements.

Figure 2 shows runtime differences in milllisecnds between Naive algorithm and algorithm we presented from Bokal et al.

Figure 2: Runtimes (in millseconds) of Naive and Bokal et. al

5.2 Diameter

Figure 3: Caption...

Now the dataset is a random walk starting at the origin. At each step we uniform randomly pick a direction, and move 0.3 distance in that direction. We keep doing this until we have enough points for an experiment.

Figure 3 shows boundaries of points that satisfy all points in range $[i, j]$ fall into some circle with radius 1.

Figure 4 will show results comparing runtimes of Naive algorithm with algorithm we presented from Bokal et al.

Figure 4: Runtimes (in millseconds) of Naive and Chan, Prat [Similar to Figure 2

6 Conclusion

This was a great project!

References

[1] Drago Bokal, Sergio Cabello, and David Eppstein. Finding All Maximal Subsequences with Hereditary Properties. In Lars Arge and János Pach, editors, *31st International Symposium on Computational Geometry (SoCG 2015)*, volume 34 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 240–254, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[2] Timothy M. Chan and Simon Pratt. Two Approaches to Building Time-Windowed Geometric Data Structures. In Sándor Fekete and Anna Lubiw, editors, *32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:15, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.