Undergraduate research opportunities

Alexandru Popa
http://alexpopa.neocities.org

April 14, 2017

1 Project 1: Implementing and comparing the SCS approximation algorithms

Introduction

The goal of the computer science discipline was, from the beginning, to provide solutions to algorithmic problems. Unfortunately, a lot of important problems with applications in areas like bioinformatics, network design and string processing are NP-hard. Therefore, under the strongly believed conjecture that $P \neq NP$, no polynomial time exact algorithms can be found for these problems. Since, finding exact solutions to these problems is not feasible, several relaxations were proposed: either the algorithms are allowed to run in exponential time or they may return suboptimal solutions (and run in polynomial time).

One *very important* NP-hard problem is the *shortest common superstring* problem described next. Since the DNA is a sequence of letters over the alphabet A, C, G, T, many problems from bioinformatics can be represented as string problems. A classical example is the *shortest common superstring* (SCS) problem, in which we are given a set of strings, and the goal is to find the shortest string that contains all the input strings as substrings. The SCS problem has a variety of crucial applications such as genome assembly, data compression and scheduling.

Problem definition

Shortest common superstring is subject of intensive research and approximation algorithms with the following ratios are shown: 3 [4] , $2\frac{8}{9}$ [15] , $2\frac{5}{6}$ [6] , $2\frac{50}{63}$ [10] , $2\frac{3}{4}$ [1] , $2\frac{50}{69}$ [2] , $2\frac{2}{3}$ [3], $2\frac{25}{42}$ [5] , $2\frac{1}{2}$ [14] [8] [13] , $2\frac{11}{23}$ [11]. There is also a natural greedy algorithm which is conjectured to have an approximation ratio of 2: select two strings with largest overlap (breaking ties arbitrarily) and replace them with their merge; the algorithm stops when there is only one string left and, obviously, this string is a superstring of all strings. On the other hand, the following lower bounds are known. For arbitrary number of input strings the problem is known to be *NP-Complete* [7] and hard to approximate within $1\frac{1}{332}$ [9] . Even for the case of binary alphabet Ott [12] presented approximation ratio lower bounds.

The specific task of the student The approximation algorithms for the SCS have been studied from the theoretical perspective. However, so far these algorithms were not implemented and compared on real data sets. The task of the student is to read the literature, understand these algorithms, implement them and compare them on sets of real data.

2 Project 2: Designing new algorithms for SCS and related problems

Introduction

The goal of the computer science discipline was, from the beginning, to provide solutions to algorithmic problems. Unfortunately, a lot of important problems with applications in areas like bioinformatics, network design and string processing are NP-hard. Therefore, under the strongly believed conjecture that $P \neq NP$, no polynomial time exact algorithms can be found for these problems. Since, finding exact solutions to these problems is not feasible, several relaxations were proposed: either the algorithms are allowed to run in exponential time or they may return suboptimal solutions (and run in polynomial time).

One *very important* NP-hard problem is the *shortest common superstring* problem described next. Since the DNA is a sequence of letters over the alphabet A, C, G, T, many problems from bioinformatics can be represented as string problems. A classical example is the *shortest common superstring* (SCS) problem, in which we are given a set of strings, and the goal is to find the shortest string that contains all the input strings as substrings. The SCS problem has a variety of crucial applications such as genome assembly, data compression and scheduling.

Problem definition

Shortest common superstring is subject of intensive research and approximation algorithms with the following ratios are shown: 3 [4] , $2\frac{8}{9}$ [15] , $2\frac{5}{6}$ [6] , $2\frac{50}{63}$ [10] , $2\frac{3}{4}$ [1] , $2\frac{50}{69}$ [2] , $2\frac{2}{3}$ [3], $2\frac{25}{42}$ [5] , $2\frac{1}{2}$ [14] [8] [13] , $2\frac{11}{23}$ [11]. There is also a natural greedy algorithm which is conjectured to have an approximation ratio of 2: select two strings with largest overlap (breaking ties arbitrarily) and replace them with their merge; the algorithm stops when there is only one string left and, obviously, this string is a superstring of all strings. On the other hand, the following lower bounds are known. For arbitrary number of input strings the problem is known to be *NP-Complete* [7] and hard to approximate within $1\frac{1}{332}$ [9] . Even for the case of binary alphabet Ott [12] presented approximation ratio lower bounds.

The specific task of the student Despite significant advances, there are still many open problems in the literature. The task of the student is to design new algorithms for several variations of SCS. Specific tasks will be discussed with the students who choose this topic.

References

[1] Chris Armen and Clifford Stein. A 2 3/4-approximation algorithm for the shortest superstring problem. Technical report, Dartmouth College, Hanover, NH, USA, 1994.

[2] Chris Armen and Clifford Stein. Improved length bounds for the shortest superstring problem (extended abstract). In *WADS*, pages 494–505, 1995.

[3] Chris Armen and Clifford Stein. A 2 2/3-approximation algorithm for the shortest superstring problem. In *CPM*, pages 87–101, 1996.

[4] Avrim Blum, Tao Jiang, Ming Li, John Tromp, and Mihalis Yannakakis. Linear approximation of shortest superstrings. In *STOC*, pages 328–336, 1991.

[5] Dany Breslauer, Tao Jiang, and Zhigen Jiang. Rotations of periodic strings and short superstrings. *Journal of Algorithms*, 24:340–353, 1996.

[6] Artur Czumaj, Leszek Gasieniec, Marek Piotrw, and Wojciech Rytter. Parallel and sequential approximation of shortest superstrings. In *SWAT*, volume 824, pages 95–106, 1994.

[7] M. R. Garey and D. S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness.* W. H. Freeman, 1979.

[8] Haim Kaplan, Moshe Lewenstein, Nira Shafrir, and Maxim Sviridenko. Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. *J. ACM*, 52(4):602–626, 2005.

[9] Marek Karpinski and Richard Schmied. Improved lower bounds for the shortest superstring and related problems. *arXiv:1111.5442*, 2011.

[10] S.R. Kosaraju, J.K. Park, and C. Stein. Long tours and short superstrings. In *FOCS*, pages 166–177, 1994.

[11] Marcin Mucha. Lyndon words and short superstrings. In *SODA*, pages 958–972, 2013.

[12] Sascha Ott. Lower bounds for approximating shortest superstrings over an alphabet of size 2. In *WG*, pages 55–64, 1999.

[13] Katarzyna E. Paluch, Khaled M. Elbassioni, and Anke van Zuylen. Simpler approximation of the maximum asymmetric traveling salesman problem. In *STACS*, pages 501–506, 2012.

[14] Z. Sweedyk. A $2\frac{1}{2}$-approximation algorithm for shortest superstring. *SIAM J. Comput.*, 29(3):954–986, 1999.

[15] Shang-Hua Teng and F. Yao. Approximating shortest superstrings. In *FOCS*, pages 158–165, 1993.