Report from Dagstuhl Seminar 18281

Synergies between Adaptive Analysis of Algorithms, Parameterized Complexity, Compressed Data Structures and Compressed Indices

Edited by

Jérémy Barbay[1], Johannes Fischer[2], Stefan Kratsch[3], and Srinivasa Rao Satti[4]

1 **University of Chile – Santiago de Chile, CL,** `jeremy@barbay.cl`
2 **TU Dortmund, DE,** `johannes.fischer@cs.tu-dortmund.de`
3 **HU Berlin, DE,** `kratsch@informatik.hu-berlin.de`
4 **Seoul National University, KR,** `ssrao10@gmail.com`

Abstract

From the 8th of July 2018 to the 13th of July 2018, a Dagstuhl Seminar took place with the topic "Synergies between Adaptive Analysis of Algorithms, Parameterized Complexity, Compressed Data Structures and Compressed Indices". There, 40 participants from as many as 14 distinct countries and four distinct research areas, dealing with running time analysis and space usage analysis of algorithms and data structures, gathered to discuss results and techniques to "go beyond the worst-case" for classes of structurally restricted inputs, both for (fast) algorithms and (compressed) data structures. The seminar consisted of (1) a first session of personal introductions, each participant presenting his expertise and themes of interests in two slides; (2) a series of four technical talks; and (3) a larger series of presentations of open problems, with ample time left for the participants to gather and work on such open problems.

Seminar July 8–13, 2018 – http://www.dagstuhl.de/18281
2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness, Theory of computation → Dynamic programming, Theory of computation → Data compression, Theory of computation → Sorting and searching, Theory of computation → Fixed parameter tractability
Keywords and phrases adaptive (analysis of) algorithms, compressed data structures, compressed indices, parameterized complexity.
Digital Object Identifier 10.4230/DagRep.8.7.44
Edited in cooperation with Falko Hegerfeld

1 Executive Summary

Jérémy Barbay (University of Chile – Santiago de Chile, CL)
Johannes Fischer (TU Dortmund, DE)
Stefan Kratsch (HU Berlin, DE)
Srinivasa Rao Satti (Seoul National University, KR)

Seminar 18281, about the "Synergies between Adaptive Analysis of Algorithms, Parameterized Complexity, Compressed Data Structures and Compressed Indices", gathered researchers

Synergies between Adaptive Analysis of Algorithms, Parameterized Complexity, Compressed Data Structures and Compressed Indices, *Dagstuhl Reports*, Vol. 8, Issue 07, pp. 44–61
Editors: Jérémy Barbay, Johannes Fischer, Stefan Kratsch, and Srinivasa Rao Satti
DAGSTUHL REPORTS Dagstuhl Reports
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

from four distinct research areas (with some researchers having results in up to three such areas, but none in all four):
1. the area of adaptive analysis of algorithms;
2. the study of parameterized complexity of NP-hard problems;
3. the area focused on compressed data structures; and
4. the area concerned with the study of compressed indices.

Goals

The intuition behind gathering people from such diverse communities was that while all of these subareas of algorithms and data structures focus on "going beyond the worst-case" for classes of structurally restricted inputs, there has been a limited amount of interactions between them, and some results have been "discovered" twice. Therefore, the main goal of the seminar was to share knowledge and make joint progress through dedicated survey talks and plenty of time for discussions and work on open problems.

Structure

The seminar consisted of
1. a first session of personal introductions, each participant presenting his expertise and themes of interests in two slides;
2. a small series of technical talks, some organized a long time in advance, and some improvised "on demand"; and
3. a larger series of presentation of open problems, with ample time left for the participants to gather and work on such open problems.

Conclusion

Most participants concurred that they learned a lot from the seminar, and acquired new contacts to foster further collaborations. In particular, interactions between the adaptive analysis of algorithms and the study of the parameterized complexity of NP-hard problems seemed relevant to the recent development of conditional lower bounds for problems classically solved in polynomial time, an approach referred to as "Fine Grained Analysis" or "FPT in P".

Generally, it appears that the seminar struck a good balance between scheduled sessions for survey talks and presentation of open problems as well as free time for discussion and interaction. During the free time, many smaller groups got together for work on open problems or for informal presentations of more specialist topics with a smaller audience. We think that this setup, along with the longer than usual round of introductions on the first day, was very successful at bringing together the different research areas.

2 Table of Contents

3 Overview of Talks

3.1 An Introduction to the Adaptive Analysis of Algorithms

Jérémy Barbay (University of Chile – Santiago de Chile, CL)

Traditionally, algorithmic theory measures the complexity of a problem or algorithm in terms of the worst-case behavior over all inputs of a given size. However, in certain cases an improved algorithm can be obtained by considering a finer partition of the input space by a difficulty measure, sometimes down to the instance itself. This finer partition is defined through a difficulty measure, which groups the instances both by their size and by their difficulty, in order to refine the worst case analysis for both upper and lower bounds on the complexity of the instance.

This approach has been the subject of extensive work, on one hand on problems which can be solved in polynomial time, and on the other hand on NP-hard problems, which solutions are checkable in polynomial time but for which no polynomial time algorithm has been found so far. Example of such results include, for polynomial time problems, Searching in a sorted array [11, 30, 10, 31], computing the Intersection [6, 5, 4, 16, 19, 18] of sorted arrays, Merging [16, 18] sorted arrays, Sorting permutations [21, 29, 7, 3, 25, 26, 27, 17, 13] and multisets [28, 9], computing Maxima Sets [15, 22, 1, 8] and Convex Hulls [12, 24, 15, 14, 20, 2, 23, 1, 8].

References

1 P. Afshani, J. Barbay, and T. M. Chan. Instance-optimal geometric algorithms. *Journal of the ACM (JACM)*, 64(1):3:1–3:38, Mar. 2017.
2 A. M. Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters (IPL)*, 9:216–219, 1979.
3 J. Barbay, J. Fischer, and G. Navarro. LRM-trees: Compressed indices, adaptive sorting, and compressed permutations. *Theoretical Computer Science (TCS)*, 459:26–41, 2012.
4 J. Barbay, A. Golynski, J. I. Munro, and S. S. Rao. Adaptive searching in succinctly encoded binary relations and tree-structured documents. *Theoretical Computer Science (TCS)*, 2007.
5 J. Barbay and C. Kenyon. Alternation and redundancy analysis of the intersection problem. *ACM Transactions on Algorithms (TALG)*, 4(1):4:1–4:18, Mar. 2008.
6 J. Barbay, A. López-Ortiz, T. Lu, and A. Salinger. An experimental investigation of set intersection algorithms for text searching. *ACM Journal of Experimental Algorithms (JEA)*, 14, 2009.
7 J. Barbay and G. Navarro. On compressing permutations and adaptive sorting. *Theoretical Computer Science (TCS)*, 513:109–123, 2013.
8 J. Barbay and C. Ochoa. Synergistic computation of planar maxima and convex hull. In *Computing and Combinatorics – 24th International Conference, COCOON 2018, , Qing Dao, China, July 2-4, 2018, Proceedings, Proceedings*, 2018.
9 J. Barbay, C. Ochoa, and S. R. Satti. Synergistic Solutions on MultiSets. In J. Kärkkäinen, J. Radoszewski, and W. Rytter, editors, *28th Annual Symposium on Combinatorial Pattern Matching (CPM 2017)*, volume 78 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
10 R. Beigel. Unbounded searching algorithms. *SIAM Journal of Computing (SJC)*, 19(3):522–537, 1990.

11 J. L. Bentley and A. C. C. Yao. An almost optimal algorithm for unbounded searching. *Information Processing Letters (IPL)*, 5(3):82–87, 1976.

12 H. Brönnimann, J. Iacono, J. Katajainen, P. Morin, J. Morrison, and G. Toussaint. Space-efficient planar convex hull algorithms. In *Proc. Latin American Theoretical Informatics (LATIN)*, pages 494–507, 2002.

13 W. H. Burge. Sorting, trees, and measures of order. *Information and Control*, 1(3):181–197, 1958.

14 Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *GEOMETRY: Discrete and Computational Geometry*, 16, 1996.

15 T. M. Chan. Output-sensitive results on convex hulls, extreme points, and related problems. *Discrete & Computational Geometry*, 16:369–387, 1996.

16 E. Chiniforooshan, A. Farzan, and M. Mirzazadeh. Worst case optimal union-intersection expression evaluation. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Proceedings of the International Conference on Automata, Languages, and Programming (ICALP)*, volume 3580 of *Lecture Notes in Computer Science*, pages 179–190. Springer, 2005.

17 C. Cook and D. Kim. Best sorting algorithm for nearly sorted lists. *Communication of the ACM (CACM)*, 23:620–624, 1980.

18 E. D. Demaine, A. López-Ortiz, and J. I. Munro. Adaptive set intersections, unions, and differences. In *Proceedings of the* 11^{th} *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 743–752, 2000.

19 E. D. Demaine, A. López-Ortiz, and J. I. Munro. Experiments on adaptive set intersections for text retrieval systems. In *Proceedings of the 3rd Workshop on Algorithm Engineering and Experiments (ALENEX), Lecture Notes in Computer Science*, pages 5–6, Washington DC, 2001.

20 H. Edelsbrunner and W. Shi. An $O(n \log^2 h)$ time algorithm for the three-dimensional convex hull problem. *SIAM Journal of Computing (SJC)*, 20:259–277, 1991.

21 V. Estivill-Castro and D. Wood. A survey of adaptive sorting algorithms. *ACM Computing Surveys (ACMCS)*, 24(4):441–476, 1992.

22 D. G. Kirkpatrick and R. Seidel. Output-size sensitive algorithms for finding maximal vectors. In *Proceedings of the Annual Symposium on Computational Geometry (SoCG)*, pages 89–96, New York, NY, USA, 1985. ACM.

23 D. G. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm? *SIAM Journal of Computing (SJC)*, 15(1):287–299, 1986.

24 C. Levcopoulos, A. Lingas, and J. S. B. Mitchell. Adaptive algorithms for constructing convex hulls and triangulations of polygonal chains. In *Proceedings of the Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 80–89, London, UK, 2002. Springer-Verlag.

25 C. Levcopoulos and O. Petersson. Sorting shuffled monotone sequences. *Inf. Comput.*, 112(1):37–50, 1994.

26 H. Mannila. Measures of presortedness and optimal sorting algorithms. *IEEE Trans. Computers*, 34(4):318–325, 1985.

27 K. Mehlhorn. Sorting presorted files. In Springer, editor, *Proceedings of the 4th GI-Conference on Theoretical Computer Science*, volume 67 of *Lecture Notes in Computer Science*, pages 199–212, 1979.

28 J. I. Munro and P. M. Spira. Sorting and searching in multisets. *SIAM Journal on Computing (SICOMP)*, 5(1):1–8, 1976.

29 O. Petersson and A. Moffat. A framework for adaptive sorting. *Discrete Applied Mathematics (DAM)*, 59:153–179, 1995.

30 J.-C. Raoult and J. Vuillemin. Optimal unbounded search strategies. In *Proceedings of the 7th Colloquium on Automata, Languages and Programming*, pages 512–530. Springer-Verlag, 1980.

31 E. M. Reingold and X. Shen. More nearly optimal algorithms for unbounded searching, part i: the finite case. *SIAM Journal of Computing (SJC)*, 20(1):156–183, 1991.

3.2 A Little About "FPT-in-P"

Till Fluschnik (TU Berlin, DE)

We give a brief introduction into the world of FPT-in-P. First, we discuss how the concepts running-time lower bounds, fixed-parameter tractability, and kernelization (lower bounds) translate from the world of NP-hard problems into the world of polynomial-time solvable problems. Herein, we give some examples known from the lecture for the successful application of the former two concepts, or, more precisely, of polynomial-linear fixed-parameter algorithms (PL-FPT) and polynomial-size linear-time kernelizations. We then study the polynomial-time solvable Negative Weight Triangle problem. The problem does not admit a truly subcubic algorithm unless the APSP-conjecture breaks, and hence, we elaborate easy-to-get polynomial-linear fixed-parameter algorithms regarding the graph parameters maximum degree and degeneracy of the graph. Along the graph parameter hierarchy we discuss on the possibilities of other parameterizations for polynomial-linear fixed-parameter algorithms. Lastly, we discuss the Graph Diameter problem. This problem, assuming the Strong Exponential Time Hypothesis (SETH), does not admit a truly subquadratic algorithm. We discuss Graph Diameter when parameterized by the vertex cover number–from both upper and lower bounds.

References

1 Archontia C. Giannopoulou, George B. Mertzios, Rolf Niedermeier: *Polynomial fixed-parameter algorithms: A case study for longest path on interval graphs*. Theor. Comput. Sci. 689: 67–95 (2017)

2 Till Fluschnik, Christian Komusiewicz, George B. Mertzios, André Nichterlein, Rolf Niedermeier, Nimrod Talmon: *When Can Graph Hyperbolicity Be Computed in Linear Time?* WADS 2017: 397–408

3 Till Fluschnik, George B. Mertzios, André Nichterlein: *Kernelization Lower Bounds for Finding Constant-Size Subgraphs*. CiE 2018: 183–193

4 Matthias Bentert, Till Fluschnik, André Nichterlein, Rolf Niedermeier: *Parameterized Aspects of Triangle Enumeration*. FCT 2017: 96–110

5 George B. Mertzios, André Nichterlein, Rolf Niedermeier: *The Power of Linear-Time Data Reduction for Maximum Matching*. MFCS 2017: 46:1–46:14

6 Virginia Vassilevska Williams, Ryan Williams: *Subcubic Equivalences between Path, Matrix and Triangle Problems*. FOCS 2010: 645–654

7 Liam Roditty, Virginia Vassilevska Williams: *Fast approximation algorithms for the diameter and radius of sparse graphs*. STOC 2013: 515–524

8 Manuel Sorge, Mathias Weller: *The Graph Parameter Hierarchy*. https://manyu.pro/assets/parameter-hierarchy.pdf

3.3 Gems in Kernelization

Bart Jansen (TU Eindhoven, NL)

When solving a hard computational problem, the running time can often be reduced by using a preprocessing step that throws away irrelevant parts of the data which are guaranteed not to affect the final answer. Until recently, there was no good explanation for the effectiveness of preprocessing. This changed when the notion of kernelization was developed within the field of parameterized complexity. It has been called ”the lost continent of polynomial time”, since the exploration of the formal model of preprocessing captured by kernelization has led to a surprisingly rich set of techniques that can reduce the size of NP-hard problem inputs in polynomial time, without changing the answer. Using a user-defined complexity-parameter, one can also give theoretical guarantees on the amount of data reduction that is achieved. This talk gives an introduction to kernelization by showcasing some of the gems of the area: elegant preprocessing schemes built on nontrivial mathematical insights. The presented gems deal with Edge Clique Cover, Vertex Cover, and Graph Coloring.

3.4 Tutorial: Introduction to Parameterized Algorithms

Bart Jansen (TU Eindhoven, NL)

This tutorial introduces the main concepts in fixed-parameter tractability. It treats both the positive toolkit (techniques for algorithms) and the negative toolkit (techniques for hardness proofs). Examples from the positive toolkit include bounded-depth search trees, kernelization, color coding, and treewidth-based dynamic programming. When it comes to hardness proofs it covers W[1]-hardness and some kernelization lower bounds.

3.5 Adaptive Algorithms (a personal view)

Ian Munro (University of Waterloo, CA)

Joint work of Ian Munro, Sebastian Wild

We say an adaptive algorithm is one that does well in the worst case but much better on specific classes of inputs. The one pass “Huffman codes” of Knuth et al. and of Vitter, from the 1970’s and 80’s are examples of this. They modify the code for characters “on the fly”, making it shorter for frequently occurring terms as their relative frequency increases. Indeed, one can view Lempel-Ziv compression schemes as a similar example. Another frequently studied problem under an adaptive model has been sorting sequences which contain some ordered runs. The work of Wood et al., also from the ‘80’s provide well-known examples. A more recent example of such a technique is Timsort, due to Tim Peters in 2002. This talk focuses primarily on this and recent improvements due to the author and Wild (ESA 2018). The problem is to stably sort a file that contains a substantial amount of already sorted runs.

The main model of computation is a PC running Linux, though theorems are also proven. The stability constraint essentially restricts one to merging consecutive segments of the input. Timsort works by keeping a stack of (references to) segments (from left to right). Depending on their relative lengths, one either merges an adjacent pair of such segments among the 4 (or so) most recently processed, or pushes a new run onto the stack. Clearly the stack is of height at most lg n. While it was originally claimed that the method took time O(n lg n) in the worst case, it took close to a decade for a real (and complex) proof of this statement.

We give a couple of other algorithms, both based on near optimal binary search trees (with elements only at the leaves), both based on the work of Mehlhorn from the '70's. The first has the general approach of looking at the middle of the file and determining the closest sorted run. One makes the split between the left and right subtrees of the root of a near optimal binary search tree either before or after this run. Then we can simply recurse on both sides of the root and create a tree with the runs at the leaves. For sorting, we recurse to the left, merging whenever possible the leftmost pair of adjacent runs (either original or created), then sort the right subtree in the same way, before doing a final merge. (The second method is similar though it avoids the "cache unfriendly" jumping about.) The methods are shown to involve a number of moves and comparisons that are essentially n times the "entropy" of the original run lengths, which matches the lower bound (at least for comparisons). Experiments show the methods perform better than others for the problem.

3.6 Introduction to Fine-grained Complexity

Ramamohan Paturi (University of California – San Diego, US)

This talk is a brief introduction to fine-grained complexity. The talk delineates the ingredients of the fine-grained complexity theory including the notion of fine-grained reductions, complexity conjectures, ETH and SETH and presents a sample of reductions among problems.

3.7 String Attractors

Nicola Prezza (University of Pisa, IT)

Joint work of Dominik Kempa, Gonzalo Navarro, Alberto Policriti, Nicola Prezza, Eva Rotenberg
Main reference Dominik Kempa, Nicola Prezza: "At the roots of dictionary compression: string attractors", in Proc. of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018, pp. 827–840, ACM, 2018.
URL http://dx.doi.org/10.1145/3188745.3188814

In the field of lossless text compression, it is known that high-order entropy is a weak model when the input contains long repetitions. Motivated by this fact, decades of research have generated myriads of so-called dictionary compressors: algorithms able to reduce the text's size by exploiting its repetitiveness (Lempel-Ziv 77 and the run-length Burrows-Wheeler transform are probably the most successful and known tools of this kind). In this (still at its outset) work, described in a preliminary series of papers [1, 2, 3], we introduce a new combinatorial object unifying dictionary compression techniques under a single theory. Our core result is that dictionary compressors are different approximations to the same, elegant, combinatorial problem: to find a small set of positions capturing all distinct text's substrings. We call such a set a string attractor. String attractors raise a number of very interesting algorithmic and combinatorial questions. In this talk we give the answer to some of these questions and present a (partial) list of exciting open problems to encourage further research on this new promising topic.

To start with, we show reductions between dictionary compressors and string attractors. This gives us the approximation ratios of dictionary compressors with respect to the smallest string attractor and allows us to solve several open problems related to the asymptotic relations between the output sizes of different dictionary compressors. We then show that k-attractor problem – that is, deciding whether a text has a size-t set of positions capturing all substrings of length at most k – is NP-complete for $k \geq 3$. We provide several approximation techniques for the smallest k-attractor, show that the problem belongs to the APX class for constant k, and give strong inapproximability results.

From the algorithmic side, we first show that string attractors provide a universal framework for compressed computation: we can design compressed data structures based on string attractors that are universal in the sense that, as implied by our reductions, can be built on top of any dictionary compressor. In particular, we give an optimal random-access data structure [1] and a universal compressed self-index [3]. We also provide an elegant characterization of string attractors based on suffix trees [2]. This characterization leads to very efficient algorithms for a range of problems: we show how to check the validity and minimality of a k-attractor in near-optimal time and how to quickly compute exact and approximate solutions. For example, we prove that a minimum 3-attractor can be found in optimal linear time on small (yet super-constant) alphabets, and a 2.45-approximation can be computed in linear time on general alphabets.

Our preliminary work leaves plenty of exciting open problems. We still do not know whether a constant approximation to the smallest k-attractor can be computed in polynomial time for general k, or what is the best approximation rate computable in polynomial time for the 3-attractor problem (in [1] we give a lower bound of 11809/11808 and an upper bound of 1.95). Moreover, we have not yet been able to assign the 2-attractor problem its complexity class (although we suspect it to be in P). Perhaps the most interesting information-theoretic question is the relation between the smallest attractor's size γ^* and the Kolmogorov complexity of the string: can we represent the string within $O(\gamma^*)$ space? can we design better compressors based on string attractors? Finally, it would be interesting to extend the concept of string attractor to infinite strings and to more complex objects such as multi-dimensional grids and graphs.

References

1 Dominik Kempa and Nicola Prezza. *At the Roots of Dictionary Compression: String Attractors.* Proceedings of the 50th Annual ACM Symposium on the Theory of Computing (STOC) June 25-29, 2018 in Los Angeles, CA.

2 Dominik Kempa, Alberto Policriti, Nicola Prezza, and Eva Rotenberg. *String Attractors: Verification and Optimization.* To appear in Proceedings of ESA 2018.

3 Gonzalo Navarro and Nicola Prezza. *Universal Compressed Text Indexing.* arXiv preprint arXiv:1803.09520 (2018).

3.8 Introduction to Generating Functions

Mireille Regnier (Ecole Polytechnique – Palaiseau, FR)

Main reference Philippe Flajolet, Robert Sedgewick: "Analytic Combinatorics", Cambridge University Press, 2009
URL https://doi.org/10.1017/CBO9780511801655

This talk presented a short introduction to «Analytic combinatorics» with a focus on the algebraic methods. The principle is to translate automatically a recursive definition of a combinatorial data structure into a functional equation satisfied by the so-called generating functions. In his seminal talk at ICALP, Ph. Flajolet compared the method to a train organisation, but a comparison with a Lego construction was made as well.

A few examples are given. Besides the classical word or binary tree enumeration, the prefix normal words example arose from the open problem introduced by Z. Lipták during the Dagstuhl seminar.

4 Working groups

4.1 BIRT – Binary IRTs for Genome String Compression

Stefan Böttcher (Universität Paderborn, DE)

The Burrows-Wheeler Transform (BWT) is one of the preferred data structures to store and search huge amounts of string data – as it is e.g. the case for genome data. However, genome data can become so huge that the space consumed by a BWT of that data becomes a bottleneck for efficient in-memory search of the BWT. A possible way out could be to find a highly compressed representation of such a BWT that can be constructed and searched in smaller memory.

The genome data sets that we want to compress consist of a huge amount of very long strings consisting of just the 4 letters 'A','C','G', and 'T' plus a small amount of escape characters '\$'. We expect to have a high repetition rate of long string patterns in our genome data. Therefore, we also expect to have extremely long runs in the BWT of this genome data. As a consequence, the number of runs is significantly smaller than the length of the BWT. That is why it is desirable to design a compressed encoding of a genome data BWT that has a size in the order of the number of runs, instead of a BWT that has a size in the order of the BWT. This problem description has been given by Travis Gagie [1]. Furthermore, if it is possible to encode a BWT in a compressed data structure that has a size in the order of the number of runs, such a compressed BWT is likely to fit into the main memory – in comparison to the complete genome data set which exceeds main memory space.

Inspired by this problem description, we have developed BIRT (=Binary Indexed Reversible Transformation) an approach of how to reduce the BWT size. Our starting point is a sequence of given strings of the alphabet {A,C,G,T,\$}. The key compression idea of BIRT consists of a number of lossless mapping steps, where each mapping step transforms one representation of the given string sequence into an equivalent representation of the same string sequence, finally yielding a compressed data structure.

First, BIRT transforms a set of given strings of the alphabet {A,C,G,T,$} into a set of strings of the alphabet {A,C,G,T} plus an extra index for the string delimiters (i.e., the $-symbols) – assuming that there is just a small number of string delimiters in comparison to the total length of the strings.

Second, BIRT transforms strings of the smaller alphabet {A,C,G,T} into binary strings of the alphabet {0,1}. For example, a sub-string CATG can be encoded as 01 00 11 10.

Third, BIRT transforms these binary strings into an IRT [2] [3], a variant of the BWT. Thereby, the IRT of these binary strings is also a binary string only. Furthermore, BIRT uses an external index to store the positions of the $-symbols in the IRT, or the end of the encoded strings respectively. In comparison to a direct encoding of an ACGT-string into a BWT, this binary encoding induces the following challenges. It does not only lead to a binary IRT being twice as long as the previously given ACGT-IRT which doubles the number of LF mapping steps needed for decoding. It also requires an additional technique to distinguish the valid rotations in the IRT from the non-valid rotations.

Fourth, BIRT includes such an additional technique that distinguishes the valid rotations in the IRT from the non-valid rotations. For example, it guarantees that a rotation ... 01 00 11 10... can only be interpreted as ...CATG..., and avoids an interpretation of a rotation ... 0 10 01 11 0 ... as ...GCT... .

Fifth, BIRT maps the binary IRT representation to a run length encoding plus a shorter binary IRT representation, encoding just one bit per run.

Sixth, as the shorter binary IRT representation only alternates 0-runs and 1-runs, it has no essential information and can be deleted without loss of information. To simplify the remaining mapping and encoding, BIRT assumes that the first run of the shorter binary IRT representation always is a 0-run – which has the length 0 if the shorter binary IRT representation starts with a 1-bit.

As a consequence of BIRT's six mapping steps, the encoding of a sequence of extremely long genome strings can be reduced to storing two data structures: an index for the positions of string delimiters $, and the run-length encoding of the binary IRT representation. Both data structures should be very small in comparison to the given genome data, as we expect to have few word delimiters ($) in comparison to letters (A,C,G,T) and as we expect to have very long runs. For the given reasons, we expect that BIRT's compression approach provides a huge possible compression ratio for genome data. We are currently implementing the BIRT compression approach and plan to evaluate it thereafter.

References

1 Travis Gagie. Problem description given at the Dagstuhl-Seminar 18281 “Synergies between Adaptive Analysis of Algorithms, Parameterized Complexity, Compressed Data Structures and Compressed Indices”, Dagstuhl Germany, July 8–13, 2018.

2 Stefan Böttcher, Alexander Bültmann, Rita Hartel. Method and Device to determine parts of a compressed string. Patent: PCT/EP2009/067499

3 Stefan Böttcher, Alexander Bültmann, Rita Hartel: Search and Modification in Compressed Texts. Data Compression Conference, Snowbird, Utah, USA, 2011.

4.2 Conditional Lower Bounds for Adaptive (Analysis of) Algorithms

Jérémy Barbay (University of Chile – Santiago de Chile, CL)

The Adaptive (analysis of) Algorithms traditionally deals with problems for which solid lower bounds are known on the computational complexity (e.g. sorting), if only because, given the freedom given by the choice of the parameters of the analysis, it is important to be able to show that one algorithm "optimally" takes advantage of the parameter.

Yet, recently the Computational Complexity of problems (such as Orthogonal Vectors, Frechet Distance, Edit Distance which do not have such "solid" lower bounds so far) was given some conditional lower bound by reducing some NP-hard problems to exponentially long instances of such problems; and one has been able to show "adaptive" results on such problems (again, Orthogonal Vectors, Frechet Distance, various types of Edit Distance) by adding parameters to the analysis of dynamic programs resolving them.

This yields the following open questions:

1. Can one refine the conditional lower bound on the complexity of Insert Swap Edit Distance (IS) to the worst case for n and k fixed, where k is one parameter which can make instances easier?
2. Can one refine the conditional lower bound on the complexity of Delete Insert Replace Edit Distance (DIR) to the worst case for n and k fixed, where k is one parameter which can make instances easier?
3. Can one refine the conditional lower bound on the complexity of Delete Insert Edit Distance (DI) to the worst case for n and k fixed, where k is one parameter which can make instances easier?
4. Can one refine the conditional lower bound on the complexity of Delete Replace Edit Distance (DR) to the worst case for n and k fixed, where k is one parameter which can make instances easier?
5. Can one refine the conditional lower bound on the complexity of Discrete Frechet Distance to the worst case for n and k fixed, where k is one parameter which can make instances easier?
6. Can one refine the conditional lower bound on the complexity of Frechet Distance to the worst case for n and k fixed, where k is one parameter which can make instances easier?
7. Can one refine the conditional lower bound on the complexity of Orthogonal Vectors to the worst case for n and k fixed, where k is one parameter which can make instances easier?

The discussion yielded various references, in particular one with a conditional lower bound on the computational complexity of deciding whether a set of vectors contains two orthogonal vectors, which is parameterized by the sum of the hamming weights of the vectors (in addition to usual parameters such as the number of vectors and the dimension).

4.3 Open Problems about Adaptive Dynamic Programming

Jérémy Barbay (University of Chile – Santiago de Chile, CL)

Until recently, there were some problems (e.g. Edit Distances, Discrete Frechet Distance, etc.) whose computational complexity was known to be polynomial (e.g. within $O(n^2)$), but for which no matching lower bound was known (e.g. the best known lower bound was $\Omega(n)$, arguing that any exact algorithm must read the whole input). In 2004, Ryan Williams [16, 17] proved the first conditional lower bound for such a problem, by reducing Satisfiability to deciding if a set contains two Orthogonal Vectors on an exponentially long instance. Since then, similar results have been proven for the computation of the Discrete Frechet Distance [7], of various Edit Distances between strings [2], of the Longest Common Subsequence [1] etc. On the other hand, recently Barbay and Pérez-Lantero proved some new parameterized results on the computation of the Insert Swap Edit Distance [5, 6] between strings, and Barbay proved some preliminary results pointing to similar parametrization for the computation of the Discrete Frechet Distance [3] between sequences of points, and of various other Edit Distances between strings [4]. This combination of results suggests the possibility of adding parameters to the analysis of the computational complexities of such problems as Orthogonal Vectors, Discrete Frechet Distance, Edit Distance, Longest Common Subsequence, which are all within $O(n^2)$, but for which matching conditional lower bounds have been shown only recently, and to measure the practicality of each such parametrization, for each problem.

References

1 A. Abboud, A. Backurs, and V. V. Williams. Tight hardness results for LCS and other sequence similarity measures. In V. Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 59–78. IEEE Computer Society, 2015.

2 A. Abboud, T. D. Hansen, V. V. Williams, and R. Williams. Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made. In D. Wichs and Y. Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 375–388. ACM, 2016.

3 J. Barbay. Adaptive Computation of the Discrete Fréchet Distance. 2018.

4 J. Barbay and A. Olivares. Indexed Dynamic Programming to boost Edit Distance and LCSS Computation. 2018.

5 J. Barbay and P. Pérez-Lantero. Adaptive computation of the swap-insert correction distance. In *Proceedings of the Annual Symposium on String Processing and Information Retrieval (SPIRE)*, pages 21–32, 2015.

6 J. Barbay and P. Pérez-Lantero. Adaptive computation of the swap-insert correction distance. In *ACM Transactions on Algorithms (TALG)*, 2018. Accepted on [2018-05-25 Fri], to appear.

7 K. Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, FOCS '14, pages 661–670, Washington, DC, USA, 2014. IEEE Computer Society.

8 E. D. Demaine, A. Lincoln, Q. C. Liu, J. Lynch, and V. V. Williams. Fine-grained I/O complexity via reductions: New lower bounds, faster algorithms, and a time hierarchy. In

A. R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 34:1–34:23. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

9 V. Estivill-Castro and D. Wood. A survey of adaptive sorting algorithms. *ACM Computing Surveys (ACMCS)*, 24(4):441–476, 1992.

10 J. Flum and M. Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

11 M. Lewenstein, S. Pettie, and V. V. Williams. Structure and hardness in P (dagstuhl seminar 16451). *Dagstuhl Reports*, 6(11):1–34, 2016.

12 A. Lincoln, V. V. Williams, J. R. Wang, and R. R. Williams. Deterministic time-space trade-offs for k-sum. In I. Chatzigiannakis, M. Mitzenmacher, Y. Rabani, and D. Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 58:1–58:14. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016.

13 A. Moffat and O. Petersson. An overview of adaptive sorting. *Australian Computer Journal (ACJ)*, 24(2):70–77, 1992.

14 O. Petersson and A. Moffat. A framework for adaptive sorting. *Discrete Applied Mathematics (DAM)*, 59:153–179, 1995.

15 J. R. Wang and R. R. Williams. Exact algorithms and strong exponential time hypothesis. In *Encyclopedia of Algorithms*, pages 657–661. 2016.

16 R. Williams. A new algorithm for optimal constraint satisfaction and its implications. *Electronic Colloquium on Computational Complexity (ECCC)*, (032), 2004.

17 R. Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348:357–365, 2005.

18 R. Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Comput. Sci.*, 348(2):357–365, Dec. 2005.

19 R. Williams and H. Yu. Finding orthogonal vectors in discrete structures. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 1867–1877, Philadelphia, PA, USA, 2014. Society for Industrial and Applied Mathematics.

20 V. V. Williams. Rna-folding – from hardness to algorithms. In P. Faliszewski, A. Muscholl, and R. Niedermeier, editors, *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22-26, 2016 – Kraków, Poland*, volume 58 of *LIPIcs*, pages 5:1–5:1. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016.

21 V. V. Williams. Fine-grained algorithms and complexity. In B. Kimelfeld and Y. Amsterdamer, editors, *21st International Conference on Database Theory, ICDT 2018, March 26-29, 2018, Vienna, Austria*, volume 98 of *LIPIcs*, pages 1:1–1:1. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

4.4 Parameterized Complexity Methodology applied to Compressed Data Structures

Jérémy Barbay (University of Chile – Santiago de Chile, CL)

Main reference Erik D. Demaine, Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, Somnath Sikdar, Blair D. Sullivan: "Structural Sparsity of Complex Networks: Random Graph Models and Linear Algorithms", CoRR, Vol. abs/1406.2587, 2014.
URL http://arxiv.org/abs/1406.2587

One of many problems used to illustrate the technique of Kernelisation in the area of Parameterized Complexity is Clique Partition, where given a graph $G = (V, E)$ one aims to partition G into a minimum number of cliques.

Given that a single clique is eminently compressible, one technique to compress a graph of user relationships (Webgraph, facebook graph, etc.) used by Navarro and Hernandez is to find an approximation of clique partition and compress the cliques independently. Not only did this lead to an efficient compression, the relatedness to cliques of the compression scheme allowed to support clique related operators (i.e. network for friends of a user).

This yields the following open questions:

- What other similar problems (Clique edit, Clique add, Clique Remove, etc.) can lead to good compression of practical graphs?
- Can polynomial parameterized algorithm be used in practice to partition optimally practical graphs?

We observed that the FPT complexities for problems on graphs are not promising: Cluster Edit has FPT complexity $O(3^k f(n))$, and Clique Partition has FPT complexity $O(2^{2^k} f(n))$. It is not clear if the benefits of an optimal partition/solution outweighs the costs compared to heuristics. It might be interesting to study FPT for partition editing in larger classes of subgraphs than Cliques (e.g. bicliques), but it is unlikely to have applications to graph compression, and a key observation might be that FPT approaches with a potential for graph compression should take advantage of specifics of the graphs considered (e.g. power laws, graphs with bounded expansion).

On the other hand, there is a hierarchy of properties on graphs already considerd in the context of FPT: bounded expansion, H-minor-free, planar graphs, graphs of bounded tree width, outerplanar graphs, forests, stars, forests and linear forests. The notion of 'graphs of bounded expansion' may be relevant to rigorously model social networks, and how algorithms can exploit the structure of social networks to provably work faster.

4.5 Adaptive Algorithms for Optimal Alphabetic Codes

Johannes Fischer (TU Dortmund, DE) and Jérémy Barbay (University of Chile – Santiago de Chile, CL)

Main reference Adriano M. Garsia, Michelle L. Wachs: "A New Algorithm for Minimum Cost Binary Trees", SIAM J. Comput., Vol. 6(4), pp. 622–642, 1977.
URL http://dx.doi.org/10.1137/0206045

The well-known Garsia-Wachs-algorithm [1] for computing the optimal alphabetic code for a given distribution on n letters takes $O(n \log n)$ time. We asked whether it is possible to identify 'structure' in the input that would lead to a faster running time, ideally $O(n)$.

References

1 Garsia A. and Wachs M. A new algorithm for minimum cost binary trees. *SIAM Journal on Computing (JCOM)*, 6(4):622–642, 1977.

4.6 Open problems on prefix normal words

Zsuzsanna Lipták (University of Verona, IT)

Joint work of Peter Burcsi, Gabriele Fici, Zsuzsanna Lipták, Frank Ruskey, Joe Sawada
Main reference Peter Burcsi, Gabriele Fici, Zsuzsanna Lipták, Frank Ruskey, Joe Sawada: "On prefix normal words and prefix normal forms", Theor. Comput. Sci., Vol. 659, pp. 1–13, 2017.
URL http://dx.doi.org/10.1016/j.tcs.2016.10.015

A binary word is called *prefix normal* if no substring has more 1s than the prefix of the same length. For example, 1101010110 is prefix normal but 1100110110 is not, because the substring 1101 has too many 1s. These words are the sequences of the first differences of the function $F(w,k) = \max\{d(u) \mid u \text{ is a substring of } w \text{ of length } k\}$, where $d(u)$ denotes the number of 1s in the binary word u. For example, for the word $w = 1100110110$, we get:

k	0	1	2	3	4	5	6	7	8	9	10
$F(w,k)$	0	1	2	2	3	4	4	4	5	6	6

The sequence of the first differences is $w' = 1101100110 =: \text{PNF}(w)$. We call this (necessarily binary) word the *prefix normal form* of w. It is prefix normal by construction, and it is the only prefix normal word in the equivalence class of w w.r.t. the equivalence $w \equiv v$ iff $F(w,\cdot) = F(v,\cdot)$.

Prefix normal words are motivated by *Binary Jumbled Pattern Matching (BJPM)*: Given a binary string T and a query (x,y), does T contain a substring with x zeros and y ones? BJPM can be solved by a linear scan of T in time $O(n)$. For the indexing variant (IBJPM), define $f(w,k)$ as the *minimum* number of 1s in a substring of w of length k. Then the answer to query (x,y) is YES if and only if $f(T,x+y) \le y \le F(T,x+y)$. Thus the problem can be solved by storing $F(w,\cdot)$ and $f(w,\cdot)$ in $O(n)$ space; this is a linear size index, with $O(1)$ query time. The current fastest computation of these functions F and f is given by Chan and Lewenstein, in $O(n^{1.89})$ time [4].

1. **Expected critical prefix length of a random prefix normal word of length n.** Let the *critical prefix* of a binary word be defined as the sum of the lengths of the first run of 1s (possibly empty) plus the first run of 0s, and $cr(w)$ be the length of w's critical prefix. E.g. $cr(1110001010) = 6$, $cr(0001100101) = 3$, $cr(1110000000) = 10$, $cr(1^n) = cr(0^n) = n$.
   Conjecture: The expected critical prefix length of a prefix normal word of length n is $O(\log n)$ [2].
   We can prove that $Exp(cr(w)) < 3$ if taken over all words of length n (for infinite words w, it is exactly 3). We can also prove that $Exp(\text{PNF}(w)) = O(\log n)$, taken over *all* binary words of length n. The paper [5] contains a table with numbers of prefix normal words for $n = 32$ and each combination of s, t, for $s = 1, ..., 7$ and $1 \le t \le n$.
2. **Equivalence class sizes.** Some equivalence classes are singletons (e.g. $1^n, 0^n, 1001, \ldots$; this implies that the word is a palindrome, since $F(w,\cdot) = F(w^{\text{rev}},\cdot)$ always), some are much larger. The OEIS sequence number A238110 [6] lists the size of the largest equivalence class for n up to 50. This question is the same as asking how many distinct words can have the same function F.

3. **Enumeration of prefix normal words.** Let $pnw(n)$ denote the number of prefix normal words of length n. It is easy to see that pnw grows exponentially. No closed form is known for $pnw(n)$; OEIS sequence number A194850 [6] lists $pnw(n)$ up to $n = 50$. We have generating functions for some (few) subsets, but not for $pnw(n)$. Asymptotic bounds exist, which seem to imply $pnw(n) = 2^{n-\Theta(\log^2 n)}$ [3].
The question can be rephrased as: How many different functions F can exist, where a necessary and sufficient condition for a 0-1 step function F to be the $F(w, \cdot)$ of some word w is that for all $i < j, F(i+j) \leq F(i) + F(j)$.
4. **Testing.** The best algorithm to decide whether a string w is prefix normal is: Compute $\text{PNF}(w)$; w is prefix normal iff $\text{PNF}(w) = w$. The fastest algorithm for doing this is given in [4]. However, it is not clear that *recognition* is as hard as computing the F-function.
5. **Which prefix normal forms w.r.t. 1 can be combined with which prefix normal forms w.r.t. 0?** Define $F_0(w, \cdot)$ and $\text{PNF}_0(w)$ analogously to above, but w.r.t. 0 instead of 1. (For constructing $\text{PNF}_0(w)$, we put a 0 when F_0 increases, and a 1 otherwise.) Then the two prefix normal forms of w encode the index for BJPM. These can be used to answer BJPM queries as follows:

$$(x, y) \text{ is a YES-query} \Leftrightarrow \text{rank}_1(\text{PNF}_0(w), x+y) \leq y \leq \text{rank}_1(\text{PNF}_1(w), x+y).$$

Prefix normal words w.r.t. 0 are defined analogously to prefix normal words w.r.t. 1. Given w, a prefix normal word w.r.t. 1, and w', a prefix normal word w.r.t. 0, we call w and w' *compatible* if there exists a binary word v s.t. $w = \text{PNF}_1(v)$ and $w' = \text{PNF}_0(v)$. The open problem is: Which prefix normal words w.r.t. 1 are compatible with which prefix normal words w.r.t. 0?
6. **How big are the Parikh-equivalence classes?** Another equivalence relation is given by: w Parikh-equivalent to v iff $\text{PNF}_1(w) = \text{PNF}_1(v)$ and $\text{PNF}_0(w) = \text{PNF}_0(v)$. Note that this holds iff the Parikh sets of w and v are the same, where the *Parikh set* of a string is the set of Parikh vectors of its substrings. How big are these equivalence classes? That is, how many different strings can have the same Parikh set?
Similar results about the multiset (not set) of Parikh vectors of substrings can be found in [1].

4.6.1 Progress during workshop

During the workshop, we showed that $Exp(cr(w)) = o(n)$, where w is a randomly chosen prefix normal word of length n, in particular that $Exp(cr(w)) = O(\sqrt{n \log n})$ (Rajeev Raman, Travis Gagie, Pat Nicholson).

References

1 J. Acharya, H. Das, O. Milenkovic, A. Orlitsky, S. Pan. String Reconstruction from Substring Compositions. *SIAM J. Discrete Math.*, 29(3):1340–1371, 2015.
2 P. Burcsi, G. Fici, Zs. Lipták, F. Ruskey, and J. Sawada. On combinatorial generation of prefix normal words. In Proc. of *CPM 2014*, LNCS vol. 8486, pp. 60–69, 2014.
3 P. Burcsi, G. Fici, Zs. Lipták, F. Ruskey, and J. Sawada. On prefix normal words and prefix normal forms. *Theoret. Comput. Sci.*, 659:1–13, 2017.
4 T. M. Chan and M. Lewenstein. Clustered integer 3SUM via additive combinatorics. In Proc. of *STOC 2015*, pages 31–40, 2015.
5 F. Cicalese, Zs. Lipták, and M. Rossi. Bubble-Flip – A new generation algorithm for prefix normal words. *Theoret. Comput. Sci.*, 743:38–52, 2018.
6 The Online Encyclopedia of Integer Sequences (OEIS). `http://www.oeis.org`. Sequences A194850, A238109, and A238110.

Participants

- Jérémy Barbay
University of Chile – Santiago de Chile, CL
- Philip Bille
Technical University of Denmark – Lyngby, DK
- Stefan Böttcher
Universität Paderborn, DE
- Luca Castelli Aleardi
Ecole Polytechnique – Palaiseau, FR
- Stephane Durocher
University of Manitoba – Winnipeg, CA
- Johannes Fischer
TU Dortmund, DE
- Till Fluschnik
TU Berlin, DE
- Allyx Fontaine
University of Guyane – Cayenne, FR
- Travis Gagie
Universidad Diego Portales, CL
- Simon Gog
eBay Inc – San Jose, US
- Meng He
Dalhousie University – Halifax, CA
- Falko Hegerfeld
HU Berlin, DE
- Shunsuke Inenaga
Kyushu University – Fukuoka, JP
- Bart Jansen
TU Eindhoven, NL
- Artur Jez
University of Wroclaw, PL
- Seungbum Jo
Universität Siegen, DE
- Ahmet Kara
University of Oxford, GB
- David G. Kirkpatrick
University of British Columbia – Vancouver, CA
- Christian Knauer
Universität Bayreuth, DE
- Dominik Köppl
TU Dortmund, DE
- Stefan Kratsch
HU Berlin, DE
- Florian Kurpicz
TU Dortmund, DE
- Zsuzsanna Lipták
University of Verona, IT
- Sebastian Maneth
Universität Bremen, DE
- Ian Munro
University of Waterloo, CA
- Yakov Nekrich
University of Waterloo, CA
- Patrick K. Nicholson
Nokia Bell Labs – Dublin, IE
- Yoshio Okamoto
The University of Electro-Communications – Tokyo, JP
- Ramamohan Paturi
University of California – San Diego, US
- Nicola Prezza
University of Pisa, IT
- Rajeev Raman
University of Leicester, GB
- Venkatesh Raman
Institute of Mathematical Sciences – Chennai, IN
- Srinivasa Rao Satti
Seoul National University, KR
- Mireille Regnier
Ecole Polytechnique – Palaiseau, FR
- Giovanna Rosone
University of Pisa, IT
- Raimund Seidel
Universität des Saarlandes, DE
- Jouni Sirén
University of California – Santa Cruz, US
- Tatiana Starikovskaya
ENS – Paris, FR
- Rossano Venturini
University of Pisa, IT
- Sandra Zilles
University of Regina, CA