Report from Dagstuhl Seminar 16431

Computation over Compressed Structured Data

Edited by

Philip Bille[1], Markus Lohrey[2], Sebastian Maneth[3], and Gonzalo Navarro[4]

1 **Technical University of Denmark – Lyngby, DK,** phbi@dtu.dk
2 **Universität Siegen, DE,** lohrey@eti.uni-siegen.de
3 **University of Edinburgh, GB,** smaneth@inf.ed.ac.uk
4 **University of Chile – Santiago de Chile, CL,** gnavarro@dcc.uchile.cl

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 16431 "Computation over Compressed Structured Data".

Seminar October 23–28, 2016 – http://www.dagstuhl.de/16431
1998 ACM Subject Classification Coding and Information Theory, Data Structures
Keywords and phrases algorithms on compressed structures, data compression, indexing, straight-line programs
Digital Object Identifier 10.4230/DagRep.6.10.99
Edited in cooperation with Fabian Peternek

1 Executive Summary

Philip Bille
Markus Lohrey
Sebastian Maneth
Gonzalo Navarro

The Dagstuhl Seminar "Computation over Compressed Structured Data" took place from October 23rd to 28th, 2016. The aim was to bring together researchers from various research directions in data compression, indexing for compressed data, and algorithms for compressed data. Compression, and the ability to index and compute directly over compressed data, is a topic that is gaining importance as digitally stored data volumes are increasing at unprecedented speeds. In particular, the seminar focused on techniques for compressed *structured data*, i.e., string, trees, and graphs, where compression schemes can exploit complex structural properties to achieve strong compression ratios.

The seminar was meant to inspire the exchange of theoretical results and practical requirements related to compression of structured data, indexing, and algorithms for compressed structured data. The following specific points were addressed.

Encoding Data Structures. The goal is to encode data structures with the minimal number of bits needed to support only the desired operations, which is also called the effective entropy. The best known example of such an encoding is the $2n$-bit structure that answers range minimum queries on a permutation of $[1, n]$, whose ordinary entropy is $n \log(n)$ bits. Determining the effective entropy and designing encodings that reach the effective

entropy leads to challenging research problems in enumerative combinatorics, information theory, and data structures.

Computation-Friendly Compression. Existing state-of-the-art compression schemes encode data by extensive and convoluted references between pieces of information. This leads to strong compression guarantees, but often makes it difficult to efficiently perform compressed computation. Recent developments have moved towards designing more computation-friendly compression schemes that achieve both strong compression and allow for efficient computation. Precise bounds on the worst-case compression of these schemes are mostly missing so far.

Repetitive Text Collections. Many of the largest sequence collections that are arising are formed by many documents that are very similar to each other. Typical examples arise from version control systems, collaborative editing systems (wiki), or sequencing of genomes from the same species. Statistical-compression does not exploit this redundancy. Recently, compressed indexes based on grammar-based compressors have been developed for repetitive text collections. They achieve a considerable compression, but on the downside operations are much slower.

Recompression. Recompression is a new technique that was successfully applied for the approximation of smallest string grammars and to solve several algorithmic problems on grammar-compressed strings. Recently, recompression has been extended from strings to trees. The long list of problems that were solved in a relatively short period using recompression indicates that there exist more applications of recompression.

Graph Compression. A lot of recent work deals with succinct data structures for graphs and with graph compression, in particular for web and network graphs. At the same time, simple queries such as in- and out-neighbors can be executed efficiently on these structures. There is a wide range of important open problems and future work. For instance, there is a strong need to support more complex graph queries, like for instance regular path queries, on compressed graphs.

The seminar fully satisfied our expectations. The 41 participants from 16 countries (Algiers, Canada, Chile, Denmark, Finland, France, Germany, Great Britain, Ireland, Italy, Israel, Japan, Korea, Poland, Spain, and US) had been invited by the organizers to give survey talks about their recent research related to the topic of the seminar. The talks covered topics related to compression (e.g., grammar-based compression of string, trees, and graphs, Lempel-Ziv compression), indexing of compressed data (e.g., set-intersection, longest common extensions, labeling schemes), algorithms on compressed data (e.g., streaming, regular expression matching, parameterized matching) and covered a wide range of applications including databases, WWW, and bioinformatics. Most talks were followed by lively discussions. Smaller groups formed naturally which continued these discussions later.

We thank Schloss Dagstuhl for the professional and inspiring atmosphere. Such an intense research seminar is possible because Dagstuhl so perfectly meets all researchers' needs. For instance, elaborate research discussions in the evening were followed by local wine tasting or by heated sauna sessions.

2 Table of Contents

3 Overview of Talks

3.1 Composite repetition-aware text indexing

Djamal Belazzougui (CERIST – Algiers, DZ)

Joint work of Djamal Belazzougui, Fabio Cunial, Travis Gagie, Nicola Prezza, Mathieu Raffinot
Main reference D. Belazzougui, F. Cunial, T. Gagie, N. Prezza, M. Raffinot, "Composite Repetition-Aware Data Structures", in Proc. of the Annual Symposium on Combinatorial Pattern Matching (CPM 2015), LNCS, Vol. 9133, pp. 26–39, Springer, 2015.
URL http://dx.doi.org/10.1007/978-3-319-19929-0_3

In highly repetitive strings, like collections of genomes from the same species, distinct measures of repetition all grow sublinearly in the length of the text, and indexes targeted to such strings typically depend only on one of these measures. We describe two data structures whose size depends on multiple measures of repetition at once, and that provide competitive tradeoffs between the time for counting and reporting all the exact occurrences of a pattern, and the space taken by the structure. The key component of our constructions is the run-length encoded BWT (RLBWT), which takes space proportional to the number of BWT runs: rather than augmenting RLBWT with suffix array samples, we combine it with data structures from LZ77 indexes, which take space proportional to the number of LZ77 factors, and with the compact directed acyclic word graph (CDAWG), which takes space proportional to the number of extensions of maximal repeats. The combination of CDAWG and RLBWT enables also a new representation of the suffix tree, whose size depends again on the number of extensions of maximal repeats, and that is powerful enough to support matching statistics and constant-space traversal.

3.2 Edit Distance: Sketching, Streaming and Document Exchange

Djamal Belazzougui (CERIST – Algiers, DZ)

Joint work of Djamal Belazzougui, Qin Zhang
Main reference D. Belazzougui, Q. Zhang, "Edit Distance: Sketching, Streaming and Document Exchange", in Proc. of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2016), pp. 51–60, IEEE, 2016.
URL http://dx.doi.org/10.1109/FOCS.2016.15

We show that in the document exchange problem, where Alice holds a binary string x of length n and Bob holds another binary string y of length n, Alice can send Bob a message of size $O(K(\log^2 K + \log n))$ bits such that Bob can recover x using the message and his input y if the edit distance between x and y is no more than K, and output "error" otherwise. Both the encoding and decoding can be done in time $O\ (n + \text{poly}(K))$. This result significantly improves the previous communication bounds under polynomial encoding/decoding time. We also show that in the referee model, where Alice and Bob hold x and y respectively, they can compute sketches of x and y of sizes $\text{poly}(K \log n)$ bits (the encoding), and send to the referee, who can then compute the edit distance between x and y together with all the edit operations if the edit distance is no more than K, and output "error" otherwise (the decoding). To the best of our knowledge, this is the *first* result for sketching edit distance using $\text{poly}(K \log n)$ bits. Moreover, the encoding phase of our sketching algorithm can be performed by scanning the input string in one pass. Thus our sketching algorithm also

implies the "first" streaming algorithm for computing edit distance and all the edits exactly using $\text{poly}(K \log n)$ bits of space.

3.3 Finger Search in Grammar-Compressed Strings

Philip Bille (Technical University of Denmark – Lyngby, DK)

Joint work of Philip Bille, Anders Roy Christiansen, Patrick Hagge Cording, Inge Li Gørtz
Main reference P. Bille, A. R. Christiansen, P. H. Cording, I. L. Gørtz, "Finger Search in Grammar-Compressed Strings", in Proc. of the 36th Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2016), LIPIcs, Vol. 65, pp. 36:1–36:16, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016.
URL http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2016.36

Grammar-based compression, where one replaces a long string by a small context-free grammar that generates the string, is a simple and powerful paradigm that captures many popular compression schemes. Given a grammar, the random access problem is to compactly represent the grammar while supporting random access, that is, given a position in the original uncompressed string report the character at that position. In this paper we study the random access problem with the finger search property, that is, the time for a random access query should depend on the distance between a specified index f, called the *finger*, and the query index i. We consider both a static variant, where we first place a finger and subsequently access indices near the finger efficiently, and a dynamic variant where also moving the finger such that the time depends on the distance moved is supported. Let n be the size the grammar, and let N be the size of the string. For the static variant we give a linear space representation that supports placing the finger in $O(\log N)$ time and subsequently accessing in $O(\log D)$ time, where D is the distance between the finger and the accessed index. For the dynamic variant we give a linear space representation that supports placing the finger in $O(\log N)$ time and accessing and moving the finger in $O(\log D + \log\log N)$ time. Compared to the best linear space solution to random access, we improve a $O(\log N)$ query bound to $O(\log D)$ for the static variant and to $O(\log D + \log\log N)$ for the dynamic variant, while maintaining linear space. As an application of our results we obtain an improved solution to the longest common extension problem in grammar compressed strings. To obtain our results, we introduce several new techniques of independent interest, including a novel van Emde Boas style decomposition of grammars.

3.4 Towards Graph Re-compression

Stefan Böttcher (Universität Paderborn, DE)

Re-compression of a compressed graph has the goal to find a stronger compression of the graph without full decompression of the graph. Having a tool for graph re-compression allows us to compress sub-graphs of a larger graph in parallel and to integrate several compressed sub-graphs afterwards by using the re-compression tool. While re-compression has been investigated for straight-line (SL) string-grammars and for SL binary tree-grammars, we present new ideas for the re-compression of SL tree-grammars with commutative terminal

nodes, i.e. SL tree-grammars representing partially unordered trees, and we extend these ideas to the re-compression of SL graph grammars representing graphs with labeled nodes and labeled edges. All our re-compression algorithms repetitively search a most frequent digram D and isolate and replace each digram occurrence of D by a new nonterminal N_D which is thereafter treated as a terminal. However, the re-compression approaches for strings, for ordered binary trees, for partially unordered trees, and for graphs differ in the following: what they consider a digram and how they define the key re-compression steps, i.e. counting (non-overlapping) digram occurrences and isolating digram occurrences.

3.5 Dynamic Relative Compression, Dynamic Partial Sums, and Substring Concatenation

Patrick Hagge Cording (Technical University of Denmark – Lyngby, DK)

Joint work of Philip Bille, Patrick Hagge Cording, Inge Li Gørtz, Frederik Rye Skjoldjensen, Hjalte Wedel Vildhøj, Søren Vind
Main reference P. Bille, P. H. Cording, I. L. Gørtz, F. R. Skjoldjensen, H. W. Vildhøj, S. Vind, "Dynamic Relative Compression, Dynamic Partial Sums, and Substring Concatenation", in Proc. of the 27th Int'l Symposium on Algorithms and Computation (ISAAC 2016), LIPIcs, Vol. 64, pp. 18:1–18:13, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016.
URL http://dx.doi.org/10.4230/LIPIcs.ISAAC.2016.18

Given a static reference string R and a source string S, a relative compression of S with respect to R is an encoding of S as a sequence of references to substrings of R. Relative compression schemes are a classic model of compression and have recently proved very successful for compressing highly-repetitive massive data sets such as genomes and web-data. We initiate the study of relative compression in a dynamic setting where the compressed source string S is subject to edit operations. The goal is to maintain the compressed representation compactly, while supporting edits and allowing efficient random access to the (uncompressed) source string. We present new data structures that achieve optimal time for updates and queries while using space linear in the size of the optimal relative compression, for nearly all combinations of parameters. We also present solutions for restricted and extended sets of updates. To achieve these results, we revisit the dynamic partial sums problem and the substring concatenation problem. We present new optimal or near optimal bounds for these problems. Plugging in our new results we also immediately obtain new bounds for the string indexing for patterns with wildcards problem and the dynamic text and static pattern matching problem.

3.6 Compressed Affix Tree Representations

Rodrigo Cánovas (University of Montpellier 2, FR)

Joint work of Rodrigo Canovas, Eric Rivals

The Suffix Tree, a crucial and versatile data structure for string analysis of large texts, is often used in pattern matching and in bioinformatics applications. The Affix Tree generalizes the Suffix Tree in that it supports full tree functionalities in both search directions. The

bottleneck of Affix Trees is their space requirement for storing the data structure. Here, we discuss existing representations and classify them into two categories: Synchronous and Asynchronous. We design Compressed Affix Tree indexes in both categories and explored how to support all tree operations bidirectionally. This work compares alternative approaches for compressing the Affix Tree, measuring their space and time trade-offs for different operations. Moreover, to our knowledge, this is the first work that compares all Compressed Affix Tree implementations offering a practical benchmark for this structure.

3.7 Computing and Approximating the Lempel-Ziv-77 Factorization in Small Space

Johannes Fischer (TU Dortmund, DE)

The Lempel-Ziv-77 algorithm greedily factorizes a text of length n into z maximal substrings that have previous occurrences, which is particularly useful for text compression. We review two recent algorithms for this task:

1. A linear-time algorithm using essentially only one integer array of length n in addition to the text. (Joint work with Tomohiro I and Dominik Köppl.) 2. An even more space-conscious algorithm using O(z) space, computing a 2-approximation of the LZ77 parse in O(n lg n) time w.h.p. (Joint work with Travis Gagie, Pawel Gawrychowski and Tomasz Kociumaka.)

3.8 GLOUDS: Representing tree-like graphs

Johannes Fischer (TU Dortmund, DE)

Joint work of Johannes Fischer, Daniel Peters
Main reference J. Fischer, D. Peters, "GLOUDS: Representing tree-like graphs", Journal of Discrete Algorithms, Vol. 36, pp. 39–49, Elsevier, 2016.
URL http://dx.doi.org/10.1016/j.jda.2015.10.004

The Graph Level Order Unary Degree Sequence (GLOUDS) is a new succinct data structure for directed graphs that are "tree-like," in the sense that the number of "additional" edges (w.r.t. a spanning tree) is not too high. The algorithmic idea is to represent a BFS-spanning tree of the graph (consisting of n nodes) with a well known succinct data structure for trees, named LOUDS, and enhance it with additional information that accounts for the non-tree edges. In practical tests, our data structure performs well for graphs containing up to $m = 5n$ edges, while still having competitive running times for listing adjacent nodes.

3.9 Queries on LZ-Bounded Encodings

Travis Gagie (Universidad Diego Portales, CL)

Joint work of Djamal Belazzougui, Travis Gagie, Pawel Gawrychowski, Juha Kärkkäinen, Alberto Ordóñez, Simon J. Puglisi, Yasuo Tabei
Main reference D. Belazzougui, T. Gagie, P. Gawrychowski, J. Kärkkäinen, A. Ordóñez, S. J. Puglisi, Y. Tabei, "Queries on LZ-Bounded Encodings", Data Compression Conference (DCC 2015), pp. 83–92, IEEE, 2015.
URL http://dx.doi.org/10.1109/DCC.2015.69

We describe a data structure that stores a strings in space similar to that of its Lempel-Ziv encoding and efficiently supports access, rank and select queries. These queries are fundamental for implementing succinct and compressed data structures, such as compressed trees and graphs. We show that our data structure can be built in a scalable manner and is both small and fast in practice compared to other data structures supporting such queries.

3.10 Distance and NCA labeling schemes for trees

Pawel Gawrychowski (University of Wroclaw, PL)

Joint work of Ofer Freedman, Paweł Gawrychowski, Jakub Łopuszański, Patrick K. Nicholson, Oren Weimann
Main reference O. Freedman, P. Gawrychowski, P. K. Nicholson, O. Weimann, "Optimal Distance Labeling Schemes for Trees", arXiv:1608.00212v1 [cs.DS], 2016.
URL https://arxiv.org/abs/1608.00212v1

Labeling schemes seek to assign a short label to each vertex in a graph, so that a function on two nodes (such as distance or adjacency) can be computed by examining their labels alone. This is particularly desirable in distributed settings, where nodes are often processed using only some locally stored data. Recently, with the rise in popularity of distributed computing platforms such as Spark and Hadoop, labeling schemes have found renewed interest. For the particular case of trees, the most natural functions are distance, ancestry, adjacency, and nearest common ancestor. We design improved labeling schemes for distance and nearest common ancestor.

For arbitrary distances, we show how to assign labels of $1/4\log^2 n + o(\log^2 n)$ bits to every node so that we can determine the distance between any two nodes given only their two labels. This closes the line of research initiated by Gavoille et al. [SODA '01] who gave an $O(\log^2 n)$ bits upper bound and a $1/8\log^2 n - O(\log n)$ bits lower bound, and followed by Alstrup et al. [ICALP '16] who gave a $1/2\log^2 n + O(\log n)$ bits upper bound and a $1/4\log^2 n - O(\log n)$ bits lower bound.

Next, for distances bounded by k, we show how to construct labels whose length is the minimum between $\log n + O(k\log(\log n/k))$ and $O(\log n \cdot \log(k/\log n))$. The query time in both cases is constant. We complement our upper bounds with almost tight lower bounds of $\log n + \Omega(k\log(\log n/(k\log k)))$ and $\Omega(\log n \cdot \log(k/\log n))$. Finally, we consider $(1+\varepsilon)$-approximate distances. We prove an $O(\log(1/\varepsilon)\cdot\log n)$ upper bound and a matching $\Omega(\log(1/\varepsilon)\cdot\log n)$ lower bound. This improves the recent $O(1/\varepsilon\cdot\log n)$ upper bound of Alstrup et al. [ICALP '16].

For nearest common ancestor, the known upper bounds are in the $O(\log n)$ regime. We significantly improve the constant factor, and in particular go below 2 for the case of binary trees.

3.11 Querying regular languages over sliding-windows

Danny Hucke (Universität Siegen, DE)

Joint work of Moses Ganardi, Danny Hucke, Markus Lohrey

We study the space complexity of querying regular languages over data streams in the sliding window model. The algorithm has to answer at any point of time whether the content of the sliding window belongs to a fixed regular language. A trichotomy is shown: For every regular language the optimal space requirement is either in $\Theta(n)$, $\Theta(\log n)$, or constant, where n is the size of the sliding window.

References

1 Charu C. Aggarwal. *Data Streams – Models and Algorithms*. Springer, 2007.
2 Arvind Arasu and Gurmeet Singh Manku. Approximate counts and quantiles over sliding windows. In *Proceedings of PODS 2004*, pages 286–296. ACM, 2004.
3 Brian Babcock, Mayur Datar, Rajeev Motwani, and Liadan O'Callaghan. Maintaining variance and k-medians over data stream windows. In *Proceedings of PODS 2003*, pages 234–243. ACM, 2003.
4 Ajesh Babu, Nutan Limaye, Jaikumar Radhakrishnan, and Girish Varma. Streaming algorithms for language recognition problems. *Theor. Comput. Sci.*, 494:13–23, 2013.
5 Vladimir Braverman. Sliding window algorithms. In *Encyclopedia of Algorithms*, pages 2006–2011. 2016.
6 Vladimir Braverman, Rafail Ostrovsky, and Carlo Zaniolo. Optimal sampling from sliding windows. *J. Comput. Syst. Sci.*, 78(1):260–272, 2012.
7 Michael S. Crouch, Andrew McGregor, and Daniel Stubbs. Dynamic graphs in the sliding-window model. In *Proceedings of ESA 2013*, volume 8125 of *Lecture Notes in Computer Science*, pages 337–348. Springer, 2013.
8 Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, 31(6):1794–1813, 2002.
9 Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata*. Springer, 2009.
10 Gudmund Skovbjerg Frandsen, Peter Bro Miltersen, and Sven Skyum. Dynamic word problems. *J. ACM*, 44(2):257–271, 1997.
11 Lukasz Golab and M. Tamer Özsu. Processing sliding window multi-joins in continuous queries over data streams. In *Proceedings of VLDB 2003*, pages 500–511. Morgan Kaufmann, 2003.
12 Markus Holzer and Barbara König. On deterministic finite automata and syntactic monoid size. *Theor. Comput. Sci.*, 327(3):319–347, 2004.
13 Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, Boston, Basel, Berlin, 1994.

3.12 The smallest grammar problem revisited

Danny Hucke (Universität Siegen, DE)

Joint work of Danny Hucke, Markus Lohrey, Carl Philipp Reh

In a seminal paper of Charikar et al. on the smallest grammar problem, the authors derive upper and lower bounds on the approximation ratios for several grammar-based compressors, but in all cases there is a gap between the lower and upper bound. Here we close the gaps for LZ78 and BISECTION by showing that the approximation ratio of LZ78 is $\Theta((n/\log n)^{2/3})$, whereas the approximation ratio of BISECTION is $\Theta((n/\log n)^{1/2})$.

References

1 J. Arpe and R. Reischuk. On the complexity of optimal grammar-based compression. In *Proc. DCC 2006*, pages 173–182. IEEE Computer Society, 2006.
2 J. Berstel and S. Brlek. On the length of word chains. *Inf. Process. Lett.*, 26(1):23–28, 1987.
3 K. Casel, H. Fernau, S. Gaspers, B. Gras, and M. L. Schmid. On the complexity of grammar-based compression over fixed alphabets. In *Proc. ICALP 2016*, Lecture Notes in Computer Science. Springer, 1996. to appear.
4 M. Charikar, E. Lehman, A. Lehman, D. Liu, R. Panigrahy, M. Prabhakaran, A. Sahai, and A. Shelat. The smallest grammar problem. *IEEE Trans. Inf. Theory*, 51(7):2554–2576, 2005.
5 A. A. Diwan. A new combinatorial complexity measure for languages. Tata Institute, Bombay, India, 1986.
6 L. Gasieniec, M. Karpinski, W. Plandowski, and W. Rytter. Efficient algorithms for Lempel-Ziv encoding (extended abstract). In *Proc. SWAT 1996*, volume 1097 of *Lecture Notes in Computer Science*, pages 392–403. Springer, 1996.
7 A. Jeż. Approximation of grammar-based compression via recompression. In *Proc. CPM 2013*, volume 7922 of *LNCS*, pages 165–176. Springer, 2013.
8 J. C. Kieffer and E.-H. Yang. Grammar-based codes: A new class of universal lossless source codes. *IEEE Trans. Inf. Theory*, 46(3):737–754, 2000.
9 J. C. Kieffer, E.-H. Yang, G. J. Nelson, and P. C. Cosman. Universal lossless compression via multilevel pattern matching. *IEEE Trans. Inf. Theory*, 46(4):1227–1245, 2000.
10 N. J. Larsson and A. Moffat. Offline dictionary-based compression. In *Proc. DCC 1999*, pages 296–305. IEEE Computer Society, 1999.
11 M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, Third Edition*. Springer, 2008.
12 M. Lohrey. *The Compressed Word Problem for Groups*. Springer, 2014.
13 C. G. Nevill-Manning and I. H. Witten. Identifying hierarchical strcture in sequences: A linear-time algorithm. *J. Artif. Intell. Res.*, 7:67–82, 1997.
14 W. Rytter. Application of Lempel-Ziv factorization to the approximation of grammar-based compression. *Theor. Comput. Sci.*, 302(1–3):211–222, 2003.
15 J. A. Storer and T. G. Szymanski. Data compression via textual substitution. *J. ACM*, 29(4):928–951, 1982.
16 Y. Tabei, Y. Takabatake, and H. Sakamoto. A succinct grammar compression. In *Proc. CPM 2013*, volume 7922 of *Lecture Notes in Computer Science*, pages 235–246. Springer, 2013.
17 J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inf. Theory*, 24(5):530–536, 1977.

3.13 A Space-Optimal Grammar Compression

Tomohiro I (Kyushu Institute of Technology, JP)

Joint work of Hiroshi Sakamoto, Tomohiro I, Yoshimasa Takabatake

A grammar compression is a context-free grammar (CFG) deriving a single string deterministically. For a CFG G in Chomsky normal form of n-symbol, it is known that an information-theoretic lower bound to represent G is $\lg n! + n + o(n)$ bits. Although a fully-online algorithm for constructing a succinct G of at most $n \lg(n+\sigma) + o(n \lg(n+\sigma))$ bits was proposed for the number σ of alphabets and the number n of variables, the optimization of the working space has remained open. We achieve the smallest $n \lg(n+\sigma) + o(n \lg(n+\sigma))$ bits of working space while preserving $O(N \frac{\lg n}{\lg \lg n})$ amortized compression time for the length N of the input string received so far.

3.14 Recompression

Artur Jez (University of Wroclaw, PL)

In this talk I will survey the recompression technique. It is based on applying simple compression operations (replacement of pairs of two different letters by a new letter and replacement of maximal repetition of a letter by a new symbol) applied to strings. The strings in question are given in a compressed way: each is represented as a context free grammar generating exactly one string, called SLP in the following. The operations are conceptually applied on the strings but they are actually performed directly on the compressed representation. For instance, when we want to replace ab in the string and the grammar have a production $X \to aY$ and the string generated by Y is bw, then we alter the rule of Y so that it generates w and replace Y with bY in all rules. In this way the rule is $X \to abY$ and so ab can be replaced. In this way we are interested mostly in the way the string is compressed rather than the string and its combinatorial properties.

The proposed method turned out to be surprisingly efficient and applicable in various scenarios: it can be used to test the equality of SLPs in time $O(n \log N)$, where n is the size of the SLP and N the length of the generated string. It can be also used to approximate the smallest SLP for a given string, with the approximation ratio $O(\log(n/g)($, where n is the length of the string and g the size of the smallest SLP for this string. Furthermore, it works also when the strings are given by more implicit representations: as solutions to word equations. This approach can be also generalized to trees and most of the results extend from the string to tree setting.

3.15 Linear Time String Indexing and Analysis in Small Space

Juha Kärkkäinen (University of Helsinki, FI)

The field of succinct data structures has flourished over the last 16 years. Starting from the compressed suffix array (CSA) by Grossi and Vitter (STOC 2000) and the FM-index by Ferragina and Manzini (FOCS 2000), a number of generalizations and applications of string indexes based on the Burrows-Wheeler transform (BWT) have been developed, all taking an amount of space that is close to the input size in bits. In many large-scale applications, the construction of the index and its usage need to be considered as one unit of computation. Efficient string indexing and analysis in small space lies also at the core of a number of primitives in the data-intensive field of high-throughput DNA sequencing. We report the following advances in string indexing and analysis. We show that the BWT of a string $T \in \{1, \ldots, \sigma\}^n$ can be built in deterministic $O(n)$ time using just $O(n \log \sigma)$ bits of space, where $\sigma \leq n$. Within the same time and space budget, we can build an index based on the BWT that allows one to enumerate all the internal nodes of the suffix tree of T. Many fundamental string analysis problems can be mapped to such enumeration, and can thus be solved in deterministic $O(n)$ time and in $O(n \log \sigma)$ bits of space from the input string. We also show how to build many of the existing indexes based on the BWT, such as the CSA, the compressed suffix tree (CST), and the bidirectional BWT index, in randomized $O(n)$ time and in $O(n \log \sigma)$ bits of space. The previously fastest construction algorithms for BWT, CSA and CST, which used $O(n \log \sigma)$ bits of space, took $O(n \log \log \sigma)$ time for the first two structures, and $O(n \log^{\varepsilon} n)$ time for the third, where ε is any positive constant. Contrary to the state of the art, our bidirectional BWT index supports every operation in constant time per element in its output.

3.16 Dynamic Rank and Select Structures on Compressed Sequences

Yakov Nekrich (University of Waterloo, CA)

Joint work of Gonzalo Navarro, Yakov Nekrich, Ian Munro

We consider the problem of storing a fully-dynamic string S in compressed form. Our representation supports insertions and deletions of symbols and answers three fundamental queries: access(i, S) returns the i-th symbol in S, rank$_a(i, S)$ counts how many times a symbol a occurs among the first i positions in S, and select$_a(i, S)$ finds the position where a symbol a occurs for the i-th time. Data structures supporting rank, select, and access queries are used in many compressed data structures and algorithms that work on string data.

We give an overview of previous results and describe two state-of-the-art solutions for this problem.

References

1 J. Ian Munro, Yakov Nekrich. *Compressed Data Structures for Dynamic Sequences*. ESA 2015:891–902
2 Gonzalo Navarro, Yakov Nekrich. *Optimal Dynamic Sequence Representations*. SIAM J. Comput. 43(5):1781–1806 (2014)

3.17 Efficient Set Intersection Counting Algorithm for Text Similarity Measures

Patrick K. Nicholson (Bell Labs – Dublin, IE)

Joint work of Preethi Lahoti, Patrick K. Nicholson, Bilyana Taneva
Main reference P. Lahoti, P. K. Nicholson, B. Taneva, "Efficient Set Intersection Counting Algorithm for Text Similarity Measures", in Proc. of the 19th Workshop on Algorithm Engineering & Experiments (ALENEX 2017), pp. 146–158, SIAM, 2017.
URL http://dx.doi.org/10.1137/1.9781611974768.12

Set intersection counting appears as a subroutine in many techniques used in natural language processing, in which similarity is often measured as a function of document cooccurence counts between pairs of noun phrases or entities. Such techniques include clustering of text phrases and named entities, topic labeling, entity disambiguation, sentiment analysis, and search for synonyms.

These techniques can have real-time constraints that require very fast computation of thousands of set intersection counting queries with little space overhead and minimal error. On one hand, while sketching techniques for approximate intersection counting exist and have very fast query time, many have issues with accuracy, especially for pairs of lists that have low Jaccard similarity. On the other hand, space-efficient computation of exact intersection sizes is particularly challenging in real-time.

In this paper, we show how an efficient space-time trade-off can be achieved for exact set intersection counting, by combining state-of-the-art algorithms with precomputation and judicious use of compression. In addition, we show that the performance can be further improved by combining the best aspects of these algorithms. We present experimental evidence that real-time computation of exact intersection sizes is feasible with low memory overhead: we improve the mean query time of baseline approaches by over a factor of 100 using a data structure that takes merely twice the size of an inverted index. Overall, in our experiments, we achieve running times within the same order of magnitude as well-known approximation techniques.

3.18 Grammar-based Graph Compression

Fabian Peternek (University of Edinburgh, GB)

Joint work of Sebastian Maneth, Fabian Peternek
Main reference S. Maneth, F. Peternek, "Compressing Graphs by Grammars", in Proc. of the 32nd Int'l Conf. on Data Engineering (ICDE 2016), IEEE, 2016.
URL http://dx.doi.org/10.1109/ICDE.2016.7498233

This talk considers a method for compressing graphs into a smaller graph grammar based on the RePair compression scheme. We start by defining the necessary notion of context-free hyperedge replacement grammars. We then extend RePair to graphs using this grammar formalism and discuss how the problem of finding non-overlapping occurrences appears more difficult on graphs than on strings and trees.

We also give some intuition on graphs that are difficult to compress with HR grammars and present some experimental results based on a proof-of-concept implementation.

Finally a short overview on two linear time speed-up algorithms for such compressed graph grammars is presented, namely reachability and regular path queries.

3.19 In-place longest common extensions

Nicola Prezza (University of Udine, IT)

Main reference N. Prezza, "In-Place Longest Common Extensions", arXiv:1608.05100v8 [cs.DS], 2016.
URL https://arxiv.org/abs/1608.05100v8

A Longest Common Extension (LCE) query returns the length of the longest common prefix between any two text suffixes. In this talk I present a deterministic data structure having the exact same size of the text (i.e. without additional low-order terms in its space occupancy) and supporting LCE queries in logarithmic time and text extraction in optimal time. Importantly, the structure can be built in-place: we can replace the text with the structure while using only O(1) memory words of additional space during construction. This results has interesting implications: I show the first sub-quadratic in-place algorithms to compute the LCP array and to solve the sparse suffix sorting problem, and a new in-place suffix array construction algorithm.

3.20 Indexing in repetition-aware space

Nicola Prezza (University of Udine, IT)

Joint work of Djamal Belazzougui, Fabio Cunial, Travis Gagie, Nicola Prezza, Mathieu Raffinot, Alberto Policriti
Main reference D. Belazzougui, F. Cunial, T. Gagie, N. Prezza, M. Raffinot, "Composite repetition-aware data structures", in Proc. of the 26th Ann. Symp. on Combinatorial Pattern Matching (CPM 2015), LNCS, Vol. 9133, pp. 26–39, Springer, 2015; pre-print available as arXiv:1502.05937v2 [cs.DS].
URL http://dx.doi.org/10.1007/978-3-319-19929-0_3
URL https://arxiv.org/abs/1502.05937v2

Full-text indexes based on the Lempel-Ziv factorization (LZ77) and on the run-length compressed Burrows-Wheeler transform (RLBWT) achieve strong compression rates, but their construction in small (compressed) space is still an open problem. In this talk, I present an algorithm to compute LZ77 in space proportional to the number of equal-letter runs in the Burrows-Wheeler transform. This result implies an asymptotically optimal-space construction algorithm for the lz-rlbwt index: an index combining LZ77 with RLBWT able to achieve exponential compression while supporting sub-quadratic time count and locate operations. I moreover present DYNAMIC: a C++ library implementing dynamic compressed data structures. This library has been used in conjunction with SDSL to implement all the discussed algorithms and indexes. I will conclude discussing different practical implementations of the lz-rlbwt index and presenting experimental results comparing our LZ77 factorization algorithm and the lz-rlbwt index with the state of the art.

3.21 Encoding Data Structures

Rajeev Raman (University of Leicester, GB)

Joint work of This is a survey talk based on papers by many authors.
Main reference R. Raman, "Encoding Data Structures", in Proc. of the 9th Int'l Workshop on Algorithms and Computation (WALCOM'15), LNCS, Vol. 8973, pp. 1–7, Springer, 2015.
URL http://dx.doi.org/10.1007/978-3-319-15612-5_1

Driven by the increasing need to analyze and search for complex patterns in very large data sets, the area of compressed and succinct data structures has grown rapidly in the last 10–15 years. Such data structures have very low memory requirements, allowing them to fit into the main memory of a computer, which in turn avoids expensive computation on hard disks.

This talk will focus on a topic that has become popular recently: encoding "the data structure" itself. Some data structuring problems involve supporting queries on data, but the queries that need to be supported do not allow the original data to be deduced from the queries. This presents opportunities to obtain space savings even when the data is incompressible, by extracting only the information needed to answer the queries when pre-processing the data, and then deleting the data. The minimum information needed to answer the queries is called the *effective entropy* of the problem: precisely determining the effective entropy leads to interesting combinatorial problems.

This survey talk is a slightly longer version of a a talk given in Dagstuhl seminar 16101 (Data Structures and and Advanced Models of Computation on Big Data) and was given at the request of the organizers.

3.22 A Linear Time Algorithm for Seeds Computation

Wojciech Rytter (University of Warsaw, PL)

Joint work of Tomasz Kociumaka, Marcin Kubica, Jakub Radoszewski, Wojciech Rytter, Tomasz Waleń
Main reference T. Kociumaka, M. Kubica, J. Radoszewski, W. Rytter, T. Waleń, "A linear time algorithm for seeds computation", in Proc. of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2012), pp. 1095-1112, ACM, 2012.
URL http://dl.acm.org/citation.cfm?id=2095116.2095202

A seed in a word is a relaxed version of a period. We show a linear-time algorithm computing a compact representation of linear size of all the seeds of a word (though the set of distinct seeds could be quadratic). In particular, the algorithm computes the shortest seed and the number of seeds. Our approach is based on combinatorial relations between seeds and a variant of the LZ-factorization (used here for the first time in context of seeds). In the previous papers the compact representation of seeds consisted of two independent representations based on the suffix tree of the word and the suffix tree of the reverse of the word. Our another contribution is a new compact representation of all seeds which avoids dealing with reversals of the word.

3.23 Space-efficient graph algorithms

Srinivasa Rao Satti (Seoul National University, KR)

Joint work of Sankardeep Chakraborty, Anish Mukherjee, Srinivasa Rao Satti, Venkatesh Raman

We reconsider various graph algorithms in the settings where there is only a limited amount of memory available, apart from the input representation. These settings are motivated by the need to run such algorithms on limited-memory devices, as well as to capture the essential features of some of the latest memory technologies. The talk presents a brief overview of some of the recent developments in this area, and gives pointers to some future directions.

3.24 On the Complexity of Grammar-Based Compression over Fixed Alphabets

Markus Schmid (Universität Trier, DE)

Joint work of Katrin Casel, Henning Fernau, Serge Gaspers, Benjamin Gras, Markus L. Schmid

It is shown that the shortest-grammar problem remains NP-complete if the alphabet is fixed and has a size of at least 24 (which settles an open question). On the other hand, this problem can be solved in polynomial-time, if the number of nonterminals is bounded, which is shown by encoding the problem as a problem on graphs with interval structure. Furthermore, we present an $O(3^n)$ exact exponential-time algorithm, based on dynamic programming. Similar results are also given for 1-level grammars, i.e., grammars for which only the start rule contains nonterminals on the right side (thus, investigating the impact of the "hierarchical depth" on the complexity of the shortest-grammar problem).

3.25 Compressed parameterized pattern matching

Rahul Shah (Louisiana State University – Baton Rouge, US)

Joint work of Arnab Ganguly, Rahul Shah, Sharma Thankachan
Main reference A. Ganguly, R. Shah, S. Thankachan, "Parameterized Pattern Matching – Succinctly", arXiv:1603.07457v2 [cs.DS], 2016.
URL https://arxiv.org/abs/1603.07457v2

The fields of succinct data structures and compressed text indexing have seen quite a bit of progress over the last two decades. An important achievement, primarily using techniques based on the Burrows-Wheeler Transform (BWT), was obtaining the full functionality of the suffix tree in the optimal number of bits. A crucial property that allows the use of BWT for designing compressed indexes is *order-preserving suffix links*. Specifically, the relative order between two suffixes in the subtree of an internal node is same as that of the suffixes obtained by truncating the first character of the two suffixes. Unfortunately, in many variants of the text-indexing problem, for e.g., parameterized pattern matching, 2D pattern matching, and order-isomorphic pattern matching, this property does not hold.

Consequently, the compressed indexes based on BWT do not directly apply. Furthermore, a compressed index for any of these variants has been elusive throughout the advancement of the field of succinct data structures. We achieve a positive breakthrough on one such problem, namely the *Parameterized Pattern Matching* problem.

Let T be a text that contains n characters from an alphabet Σ, which is the union of two disjoint sets: Σ_s containing static characters (s-characters) and Σ_p containing parameterized characters (p-characters). A pattern P (also over Σ) matches an equal-length substring S of T iff the s-characters match exactly, and there exists a one-to-one function that renames the p-characters in S to that in P. The task is to find the starting positions (occurrences) of all such substrings S. Previous index [Baker, STOC 1993], known as *Parameterized Suffix Tree*, requires $\Theta(n \log n)$ bits of space, and can find all *occ* occurrences in time $O(|P| \log \sigma + occ)$, where $\sigma = |\Sigma|$. We introduce an $n \log \sigma + O(n)$-bit index with $O(|P| \log \sigma + occ \cdot \log n \log \sigma)$ query time. At the core, lies a new BWT-like transform, which we call the *Parameterized Burrows-Wheeler Transform* (pBWT). The techniques are extended to obtain a succinct index for the *Parameterized Dictionary Matching* problem of Idury and Schäffer [CPM, 1994].

3.26 Streaming Pattern Matching

Tatiana Starikovskaya (University Paris-Diderot, FR)

Joint work of Raphaël Clifford, Allyx Fontaine, Ely Porat, Benjamin Sach, Tatiana Starikovskaya

In the streaming model of computation we assume that the data arrives sequentially, one data item at a time. The goal is to develop algorithms that process the data on the fly while using as little space as possible. We give a survey of recent results for the pattern matching problem in this model. We first review the main ideas behind the algorithm for exact pattern matching problem given by Porat and Porat and show how to extend them to the case of several patterns (dictionary matching). We then proceed to the approximate pattern matching problem under Hamming distance. In this problem we must compute the Hamming distance for all alignments of the given pattern and the text. We first consider the famous variant of this problem called the k-mismatch problem, where we are only interested in small Hamming distances (smaller than a given threshold k). Finally, we consider a problem of computing all Hamming distances and present an approximate algorithm for it.

References

1 R. Clifford, A. Fontaine, E. Porat, B. Sach, T. Starikovskaya. *The k-mismatch problem revisited*. SODA 2016:2039–2052. DOI: http://dx.doi.org/10.1137/1.9781611974331.ch142
2 R. Clifford, T. Starikovskaya. *Approximate Hamming Distance in a Stream*. ICALP 2016: 20:1–20:14 DOI: http://dx.doi.org/10.4230/LIPIcs.ICALP.2016.20
3 R. Clifford, A. Fontaine, E. Porat, B. Sach, T. Starikovskaya. *Dictionary Matching in a Stream*. ESA 2015:361–372 DOI: http://dx.doi.org/10.1007/978-3-662-48350-3_31

3.27 Quickscorer: a fast algorithm to rank documents with additive ensembles of regression trees

Rossano Venturini (University of Pisa, IT)

Joint work of Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, Rossano Venturini
Main reference C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, N. Tonellotto, R. Venturini, "QuickScorer: a Fast Algorithm to Rank Documents with Additive Ensembles of Regression Trees", in Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'15), pp. 73–82, ACM, 2015.
URL http://dx.doi.org/10.1145/2766462.2767733

Learning-to-Rank models based on additive ensembles of regression trees have proven to be very effective for ranking query results returned by Web search engines, a scenario where quality and efficiency requirements are very demanding. Unfortunately, the computational cost of these ranking models is high. Thus, several works already proposed solutions aiming at improving the efficiency of the scoring process by dealing with features and peculiarities of modern CPUs and memory hierarchies. In this paper, we present QuickScorer, a new algorithm that adopts a novel bitvector representation of the tree-based ranking model, and performs an interleaved traversal of the ensemble by means of simple logical bitwise operations. The performance of the proposed algorithm are unprecedented, due to its cacheaware approach, both in terms of data layout and access patterns, and to a control flow that entails very low branch mis-prediction rates. The experiments on real Learning-to-Rank datasets show that QuickScorer is able to achieve speedups over the best state-of-the-art baseline ranging from 2x to 6.5x.

4 Working groups

4.1 LZ78 Construction in Little Main Memory Space

Diego Arroyuelo (TU Federico Santa María – Valparaíso, CL), Rodrigo Cánovas (University of Montpellier 2, FR), Gonzalo Navarro (University of Chile – Santiago de Chile, CL), and Rajeev Raman (University of Leicester, GB)

Joint work of Diego Arroyuelo, Rodrigo Cánovas, Gonzalo Navarro, Andreas Poyias, Rajeev Raman
Main reference A. Poyias, R. Raman, "Improved Practical Compact Dynamic Tries", in Proc. of the 22nd Int'l Symposium on String Processing and Information Retrieval (SPIRE 2015), LNCS, Vol. 9309, pp. 324–336, Springer, 2015.
URL http://dx.doi.org/10.1007/978-3-319-23826-5_31

We report on ongoing work aiming to do the LZ78 parsing of a text T [1..n] over alphabet $[1..\sigma]$ in linear randomized time, using only $O(z \log \sigma)$ bits of main memory, while reading the input text from disk and writing the compressed text to disk. The text can also be decompressed within the same main memory usage.

4.2 Smaller Structures for Top-k Document Retrieval

Simon Gog (KIT – Karlsruher Institut für Technologie, DE), Julian Labeit, and Gonzalo Navarro (University of Chile – Santiago de Chile, CL)

Main reference J. Labeit, S. Gog, "Elias-Fano meets Single-Term Top-k Document Retrieval",in Proc. of the 19th Workshop on Algorithm Engineering & Experiments (ALENEX 2017), pp. 135–145, SIAM, 2017.
URL http://dx.doi.org/10.1137/1.9781611974768.11

In a recent paper (main reference) Labeit and Gog improve upon the space of a previous fast top-k document retrieval index by Gog and Navarro [Improved Single-Term Top-k Document Retrieval. Proc. ALENEX'15, pages 24-32], by sharply reducing the information stored about document identifiers. We now plan to further reduce the space, without hopefully affect the time too much, by removing the information on frequencies, which is the largest remaining component of the index. We plan to replace this information with a small index per document, using a previously developed (and unpublished) technique by Navarro to store many small document indexes within little space overhead.

4.3 More Efficient Representation of Web and Social Graphs by Combining GLOUDS with DSM

Cecilia Hernández Rivas (University of Concepción, CL), Johannes Fischer (TU Dortmund, DE), Gonzalo Navarro (University of Chile – Santiago de Chile, CL), and Daniel Peters

Main reference J. Fischer, D. Peters, "GLOUDS: Representing tree-like graphs", Journal of Discrete Algorithms, Vol. 36, pp. 39–49, Elsevier, 2016.
URL http://dx.doi.org/10.1016/j.jda.2015.10.004

We plan to combine the recently proposed GLOUDS representation [1] with DSM, a technique used to compress Web and social graphs by exploiting the presence of bicliques and dense subgraphs [2]. Since GLOUDS benefits from a representation with fewer edges per node and DSM reduces the number of edges from $m * n$ to $m + n$ when representing an (m, n)-biclique, we believe the combination can lead to better compression performance than the one obtained with DSM alone, and can offer reasonable edge extraction time.

References

1 J. Fischer and D. Peters. GLOUDS: Representing tree-like graphs. J. Discrete Algorithms 36:39–49 (2016).
2 C. Hernández, G. Navarro. Compressed representations for web and social graphs. Knowl. Inf. Syst. 40(2):279–313 (2014)

Participants

- Diego Arroyuelo
TU Federico Santa María – Valparaíso, CL
- Hideo Bannai
Kyushu University – Fukuoka, JP
- Djamal Belazzougui
CERIST – Algiers, DZ
- Philip Bille
Technical University of Denmark – Lyngby, DK
- Stefan Böttcher
Universität Paderborn, DE
- Rodrigo Cánovas
University of Montpellier 2, FR
- Patrick Hagge Cording
Technical University of Denmark – Lyngby, DK
- Héctor Ferrada
University of Helsinki, FI
- Johannes Fischer
TU Dortmund, DE
- Travis Gagie
Universidad Diego Portales, CL
- Adrià Gascón
University of Edinburgh, GB
- Pawel Gawrychowski
University of Wroclaw, PL
- Simon Gog
KIT – Karlsruher Institut für Technologie, DE
- Inge Li Gørtz
Technical University of Denmark – Lyngby, DK
- Cecilia Hernández Rivas
University of Concepción, CL
- Danny Hucke
Universität Siegen, DE
- Tomohiro I
Kyushu Institute of Technology, JP
- Shunsuke Inenaga
Kyushu University – Fukuoka, JP
- Artur Jez
University of Wroclaw, PL
- Juha Kärkkäinen
University of Helsinki, FI
- Susana Ladra González
University of A Coruña, ES
- Markus Lohrey
Universität Siegen, DE
- Sebastian Maneth
University of Edinburgh, GB
- Ian Munro
University of Waterloo, CA
- Gonzalo Navarro
University of Chile – Santiago de Chile, CL
- Yakov Nekrich
University of Waterloo, CA
- Patrick K. Nicholson
Bell Labs – Dublin, IE
- Alberto Ordóñez
University of A Coruña, ES
- Fabian Peternek
University of Edinburgh, GB
- Nicola Prezza
University of Udine, IT
- Rajeev Raman
University of Leicester, GB
- Wojciech Rytter
University of Warsaw, PL
- Hiroshi Sakamoto
Kyushu Institute of Technology – Fukuoka, JP
- Srinivasa Rao Satti
Seoul National University, KR
- Markus Schmid
Universität Trier, DE
- Manfred Schmidt-Schauss
Goethe-Universität – Frankfurt a. M., DE
- Rahul Shah
Louisiana State University – Baton Rouge, US
- Ayumi Shinohara
Tohoku University – Sendai, JP
- Tatiana Starikovskaya
University Paris-Diderot, FR
- Alexander Tiskin
University of Warwick – Coventry, GB
- Rossano Venturini
University of Pisa, IT