

Report from Dagstuhl Seminar 13232

# Indexes and Computation over Compressed Structured Data

Edited by

Sebastian Maneth<sup>1</sup> and Gonzalo Navarro<sup>2</sup>

1 Universität Leipzig, DE, [sebastian.maneth@gmail.com](mailto:sebastian.maneth@gmail.com)

2 University of Chile – Santiago, CL, [gnavarro@dcc.uchile.cl](mailto:gnavarro@dcc.uchile.cl)

---

## Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 13232 “Indexes and Computation over Compressed Structured Data”.

**Seminar** 3.–7. June, 2013 – [www.dagstuhl.de/13232](http://www.dagstuhl.de/13232)

**1998 ACM Subject Classification** E.1 Data Structures, E.2 Data storage representations, E.4 Coding and Information Theory: data compaction and compression

**Keywords and phrases** Compression, Indexes, Data Structures


**Digital Object Identifier** 10.4230/DagRep.3.6.22

**Edited in cooperation with** Patrick Nicholson

## 1 Executive Summary

*Sebastian Maneth*

*Gonzalo Navarro*

**License**  Creative Commons BY 3.0 Unported license  
© Sebastian Maneth and Gonzalo Navarro

The Dagstuhl Seminar “Indexes and Computation over Compressed Structured Data” took place from June 2nd to 7th, 2013. The aim was to bring together researchers from various research directions of compression and indexing of structured data. Compression, and the ability to compute directly over compressed structures, is a topic that is gaining importance as digitally stored data volumes are increasing at unprecedented speeds. Of particular interest is the combination of compression schemes with indexes that give fast access to particular operations. The seminar was meant to inspire the exchange of theoretical results and practical requirements related compression and indexing. These points were addressed in particular

- Tractability versus Intractability for Algorithmic Problems on Compressed Data
- Compression Algorithms for Strings, Trees, and Graphs
- Indexes for Compressed Data
- Algorithms for Compressed Data
- Better Search Results: Ranking and TF/IDF
- Applications of Structure Compression to other Areas

The seminar fully satisfied our expectations. The 34 participants from 11 countries (Canada, Chile, Denmark, Finland, Germany, Great Britain, Italy, Israel, Japan, 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.



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Indexes and Computation over Compressed Structured Data, *Dagstuhl Reports*, Vol. 3, Issue 6, pp. 22–37

Editors: Sebastian Maneth and Gonzalo Navarro



DAGSTUHL  
REPORTS

Dagstuhl Reports  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

grammar-based string compression) databases (e.g., XML, and top- $k$  query answering), data structures (e.g. wavelet tries), string matching, and ranged to broad application areas such as biology. 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

### Executive Summary

<i>Sebastian Maneth and Gonzalo Navarro</i> . . . . .	22
---	----

### Overview of Talks

Tree Compression with Top Trees <i>Philip Bille</i> . . . . .	26
Updates in compressed text and compressed XML <i>Stefan Boettcher</i> . . . . .	26
Grammar Indexes and Document Listing <i>Francisco Claude</i> . . . . .	27
LZ-Compressed String Dictionaries <i>Johannes Fischer</i> . . . . .	27
A Faster Grammar-Based Self-Index <i>Travis Gagie</i> . . . . .	28
(Approximate) Pattern matching in LZW-compressed texts <i>Pawel Gawrychowski</i> . . . . .	28
Algorithms on grammar based strings <i>Shunsuke Inenaga</i> . . . . .	29
Local recompression for compressed text <i>Artur Jeż</i> . . . . .	29
List Update for Data Compression <i>Alejandro López-Ortiz</i> . . . . .	30
Indexing Graphs for Path Queries with Applications in Genome Research <i>Veli Maekinen</i> . . . . .	30
XML Compression via DAGs Unranked trees can be represented using their minimal dag (directed acyclic graph) <i>Sebastian Maneth</i> . . . . .	31
Succinct Data Structures <i>J. Ian Munro</i> . . . . .	31
Indexing Highly Repetitive Collections <i>Gonzalo Navarro</i> . . . . .	32
Categorical Range Reporting <i>Yakov Nekrich</i> . . . . .	32
How to Cook a Poset <i>Patrick K. Nicholson</i> . . . . .	32
Bioinformatics Algorithms: Sequence Analysis, Genome Rearrangements, and Phylogenetic Reconstruction <i>Enno Ohlebusch</i> . . . . .	33
Wavelet Tries <i>Giuseppe Ottaviano</i> . . . . .	33

Lightweight Lempel-Ziv Parsing <i>Simon J. Puglisi</i> . . . . .	33
Encoding Top-k Queries <i>Rajeev Raman</i> . . . . .	34
Compressed Pattern Matching on Terms <i>Manfred Schmidt-Schauss</i> . . . . .	34
Top- <i>k</i> Document Retrieval <i>Rahul Shah</i> . . . . .	34
Semi-local LCS: Superglue for string comparison <i>Alexander Tiskin</i> . . . . .	35
Distributed String Mining <i>Niko Vaelimaeki</i> . . . . .	35
<b>Participants</b> . . . . .	<b>37</b>

### 3 Overview of Talks

#### 3.1 Tree Compression with Top Trees

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

**License** © Creative Commons BY 3.0 Unported license  
© Philip Bille

**Joint work of** Bille, Philip; Gørtz, Inge Li; Landau, Gad M.; Weimann, Oren

**Main reference** P. Bille, I.L. Gørtz, G.M. Landau, O. Weimann, “Tree Compression with Top Trees,” in Proc. of the 40th Int’l Colloquium on Automata, Languages, and Programming (ICALP’13), LNCS, Vol. 7965, pp. 160–171, Springer, 2013.

**URL** [http://dx.doi.org/10.1007/978-3-642-39206-1\\_14](http://dx.doi.org/10.1007/978-3-642-39206-1_14)

**URL** <http://arxiv.org/abs/1304.5702>

We introduce a new compression scheme for labeled trees based on top trees. Our compression scheme is the first to simultaneously take advantage of internal repeats in the tree (as opposed to the classical DAG compression that only exploits rooted subtree repeats) while also supporting fast navigational queries directly on the compressed representation. We show that the new compression scheme achieves close to optimal worst-case compression, can compress exponentially better than DAG compression, is never much worse than DAG compression, and supports navigational queries in logarithmic time.

#### 3.2 Updates in compressed text and compressed XML

*Stefan Boettcher (Universität Paderborn, DE)*

**License** © Creative Commons BY 3.0 Unported license  
© Stefan Boettcher

**Joint work of** Boettcher, Stefan; Buelmann, Alexander; Hartel, Rita; Schuessler, Jonathan

**URL** [http://www.cs.uni-paderborn.de/fileadmin/Informatik/AG-Boettcher/Papers\\_for\\_Download/dagstuhl2013-3.pdf](http://www.cs.uni-paderborn.de/fileadmin/Informatik/AG-Boettcher/Papers_for_Download/dagstuhl2013-3.pdf)

Compressed XML files and compressed column-oriented main memory text databases require to support insert and delete operations of the Nth word of the represented text T on a compressed version C(T) of T without full decompression of C(T). We present IRT (=Indexed Reversible Transformation), a block-sorting technique that differs from BWT by sorting word delimiters according to their occurrence in the given text T, and we present a compression technique for IRT transformed text, such that both techniques together transform a given text T into a compressed form C(T) that allows to delete the Nth word of T from C(T) and to insert a word as the Nth word of T into C(T) without full decompression of C(T). Thereby, we enable faster delete and insert operations on compressed XML files and on compressed main memory text databases. This talk is based on a conference paper [1].

#### References

- 1 Stefan Böttcher, Alexander Bültmann, Rita Hartel, Jonathan Schlußler: *Implementing efficient up-dates in compressed big text databases*. In Proc. 24th International Conference on Data-base and Expert Systems Applications (DEXA 2013), Prague, Czech Republic, (2013).

### 3.3 Grammar Indexes and Document Listing

*Francisco Claude (University of Waterloo, CA)*

**License** © Creative Commons BY 3.0 Unported license  
© Francisco Claude

**Joint work of** Munro, J. Ian; Navarro, Gonzalo;

We introduce the first grammar-compressed representation of a sequence that supports searches in time that depends only logarithmically on the size of the grammar. Given a text  $T[1..u]$  that is represented by a (context-free) grammar of  $n$  (terminal and nonterminal) symbols and size  $N$  (measured as the sum of the lengths of the right hands of the rules), a basic grammar-based representation of  $T$  takes  $N \lg n$  bits of space. Our representation requires  $2N \lg n + N \lg u + \epsilon(n \lg n) + o(N \lg n)$  bits of space, for any  $0 < \epsilon \leq 1$ . It can find the positions of the occurrences of a pattern of length  $m$  in  $T$  in  $O((m^2/\epsilon) \lg((\lg u)/(\lg n)) + (m + \text{occ}) \lg n)$  time, and extract any substring of length  $l$  of  $T$  in time  $O(l + h \lg(N/h))$ , where  $h$  is the height of the grammar tree.

We also show a practical version of this index adapted to solve the document listing problem on versioned documents. Our index is the first one based on grammar-compression. This allows for good results on repetitive collections, whereas classical techniques cannot achieve competitive space for solving the same problem. As a result of this, our index is about 16 times smaller when compared to the state of the art for document listing [Navarro, Puglisi, and Valenzuela, SEA2011]. Our query times are competitive with the state of the art.

### 3.4 LZ-Compressed String Dictionaries

*Johannes Fischer (KIT – Karlsruhe Institute of Technology, DE)*

**License** © Creative Commons BY 3.0 Unported license  
© Johannes Fischer

**Joint work of** Arz, Julian; Fischer, Johannes

**Main reference** J. Arz, J. Fischer, “LZ-Compressed String Dictionaries,” arXiv:1305.0674v1 [cs.DS], 2013.

**URL** <http://arxiv.org/abs/1305.0674v1>

We review existing compressed string dictionaries (see bibliography) and further show how to compress string dictionaries using the Lempel-Ziv (LZ78) data compression algorithm. Our approach is validated experimentally on dictionaries of up to 1.5 GB of uncompressed text. We achieve compression ratios often outperforming the existing alternatives, especially on dictionaries containing many repeated substrings. Our query times remain competitive.

Presenting the paper at Dagstuhl proved to be very fruitful, because several participants noted that if our LZ-parsing is done in reverse direction and the resulting phrases are subsequently reversed again, then it is possible to relate the size of the resulting data structures to the number of phrases in the original LZ78 parsing. Thanks a lot!

### 3.5 A Faster Grammar-Based Self-Index

*Travis Gagie (University of Helsinki, FI)*

**License** © Creative Commons BY 3.0 Unported license  
© Travis Gagie

**Joint work of** Gagie, Travis; Gawrychowski, Paweł; Karkkainen, Juha; Nekrich, Yakov; Puglisi, Simon  
**Main reference** T. Gagie, P. Gawrychowski, J. Kärkkäinen, Y. Nekrich, S.J. Puglisi, “A Faster Grammar-Based Self-Index,” arXiv:1109.3954v6 [cs.DS], 2013; randomized version submitted to “Information and Computation”.

**URL** <http://arxiv.org/abs/1109.3954v6>

To store and search genomic databases efficiently, researchers have recently started building compressed self-indexes based on grammars and LZ77. We show how, given a text  $S[1..n]$  whose LZ77 parse consists of  $z$  phrases, we can build an  $O(z \log n)$ -space deterministic index for  $S$  such that later, given a pattern  $P[1..m]$ , we can find all *occ* occurrences of  $P$  in  $S$  in  $O(m \log m + occ \log \log n)$  time.

### 3.6 (Approximate) Pattern matching in LZW-compressed texts

*Paweł Gawrychowski (MPI für Informatik – Saarbrücken, DE)*

**License** © Creative Commons BY 3.0 Unported license  
© Paweł Gawrychowski

Pattern matching is the most basic problem concerning processing text data. Its complexity seems rather well-understood, and there are quite a few very efficient algorithms that can be used to solve it. What seems not that well-understood is its compressed variant, where instead of the text (or the pattern) we are given its compressed representation, and the goal is to achieve running time depending on the size of this representation and not the original length. I will present a number of results concerning LZW-compressed pattern matching, which is pattern matching for texts compressed using Lempel-Ziv-Welch-like methods. Such methods are simple to implement, so they are used in practice, and on the other hand they are complicated enough to be interesting from the theoretical point of view. It turns out that even when both the text and the pattern are LZW-compressed, it is possible to solve pattern matching in linear time, where linear means linear in the size of the compressed representation. I will present some ideas used to prove this. Of course in practice we are more interested in approximate pattern matching, meaning pattern matching with errors or with mismatches. The previous results for approximate LZW-compressed pattern matching were based on a more or less blackbox application of some uncompressed approximate pattern matching tools, and hence took at least  $O(nm)$  time even when the bound on the number or mismatches/errors was very small. I will discuss a recent result with Straszak where we achieve  $O(n\sqrt{m})$  for constant  $k$ .

### 3.7 Algorithms on grammar based strings

*Shunsuke Inenaga (Kyushu University, JP)*

**License** © Creative Commons BY 3.0 Unported license  
© Shunsuke Inenaga

**Joint work of** Inenaga, Shunsuke; Bannai, Hideo; Masayuki, Takeda; I, Tomohiro; Gawrychowski, Paweł; Shinohara, Ayumi; Narisawa, Kazuyuki; Gagie, Travis; Lewenstein, Moshe; Landau, Gad; Goto, Keisuke; Yamamoto, Takanori; Tanaka, Toshiya; Matsubara, Wataru

Straight-line programs (SLPs) are widely accepted abstract model of outputs of grammar-based text compression algorithms. In this survey talk, I introduce our recent results on algorithms that process given SLPs efficiently. Our methods allow various operations on SLPs, such as computing q-gram frequencies, finding palindromes, squares, runs, etc. All of our algorithms do not explicitly decompress given SLPs, and run in time polynomial in the input size.

### 3.8 Local recompression for compressed text

*Artur Jeż (MPI für Informatik – Saarbrücken, DE)*

**License** © Creative Commons BY 3.0 Unported license  
© Artur Jeż

**Main reference** A. Jeż, “Faster Fully Compressed Pattern Matching by Recompression,” in Proc. of the 39th Int’l Colloquium on Automata, Languages, and Programming (ICALP’12), LNCS, Vol. 7391, pp. 533–544, Springer, 2012.

**URL** [http://dx.doi.org/10.1007/978-3-642-31594-7\\_45](http://dx.doi.org/10.1007/978-3-642-31594-7_45)

**URL** <http://arxiv.org/abs/1111.3244>

In this talk we present a simple and natural local recompression technique that is applicable to implicit representations of text, such as grammars, compressed representations or word equations. In essence the method aims at having all the strings in the instance compressed in the same way. The compression is achieved by two simple rewriting rules: pair compression that replaces all appearances of a pair of different letters  $ab$  with a fresh letter  $c$  and block compression which replaces maximal block of the form  $a^k$  with a fresh letter  $a_k$ , for all possible  $k$ . The crucial part of the method is that we can apply those rules directly to the implicit representation, however, in order to do this we may need to change this representation a bit. Our changes are local and boil down to replacement of a nonterminal (variable, piece of compressed data etc.)  $X$  with  $bX$  or  $Xa$ . With appropriate choice of pairs to compress it can be shown that the length of the strings in the instance drop by a constant factor in each phase and on the other hand the size of the instance remains linear in the input size. The method finds application in checking the equivalence of two SLPs, fully compressed pattern matching (for SLPs), word equations and approximation of the smallest grammar. In all those cases the algorithm based on recompression matches or improves the currently best known.



### 3.9 List Update for Data Compression

*Alejandro López-Ortiz (University of Waterloo, CA)*

**License** © Creative Commons BY 3.0 Unported license  
© Alejandro López-Ortiz

**Joint work of** López-Ortiz, Alejandro; Kamali, Shahin; Ladra, Susana; Seco, Diego; Dorrigiv, Reza

From inception, list update (LU) has been used as a means to compress data. In this talk we review the main practical results on the use of list update algorithms for data compression. We discuss the theoretical foundations of these results, then we present an LU-based compressing scheme which is superior to BWT. Interestingly enough this compression-inspired strategy also proves superior to MTF in the MRM cost model of Martinez, Roura and Munro in practice.

### 3.10 Indexing Graphs for Path Queries with Applications in Genome Research

*Veli Maekinen (University of Helsinki, FI)*

**License** © Creative Commons BY 3.0 Unported license  
© Veli Maekinen

**Joint work of** Sirén, Jouni; Vaelimaeki, Niko; Maekinen, Veli

**Main reference** J. Sirén, N. Välimäki, V. Mäkinen, “Indexing Finite Language Representation of Population Genotypes,” in Proc. of the 11th Int’l Workshop on Algorithms in Bioinformatics, LNCS, Vol. 6833, pp. 270–281, Springer, 2011.

**URL** [http://dx.doi.org/10.1007/978-3-642-23038-7\\_23](http://dx.doi.org/10.1007/978-3-642-23038-7_23)

We propose a generic approach to replace the canonical sequence representation of genomes with graph representations, and study several applications of such extensions. The technical tool is to extend Burrows- Wheeler transform (BWT) of strings to acyclic directed labeled graphs, to support path queries as an extension to substring searching. We develop, apply, and tailor this technique to the following applications: a) read alignment on an extended BWT index of a graph representing reference genome and known variants of it; b) split-read alignment on an extended BWT index of a splicing graph; c) read alignment on an extended BWT index of a phylogenetic tree of partial-order graphs. Other possible applications include probe/primer design and alignments to assembly graphs. The main focus in this article is on a), for which several technical and compatibility issues had to be resolved to make the approach practical. For index construction we develop a space-efficient algorithm that scales to human genome data. For queries we extend an efficient search-space pruning technique to enable approximate searches. For compatibility we tailor the approach so that alignments to paths can be projected back to the reference genome, so that the results can be seamlessly plugged inside widely adopted workflows for variation calling. Finally, we report several experiments on the feasibility of the approach to these applications. This talk is based on journal version (under preparation) of our work in WABI 2011 [1].

#### References

- 1 Jouni Sirén, Niko Välimäki, and Veli Mäkinen. *Indexing Finite Language Representation of Population Genotypes*. In Proc. WABI, LNCS 6833, pp. 270–281, Springer, 2011.

### 3.11 XML Compression via DAGs Unranked trees can be represented using their minimal dag (directed acyclic graph)

*Sebastian Maneth (University of Oxford, UK)*

**License** © Creative Commons BY 3.0 Unported license  
© Sebastian Maneth

**Joint work of** Markus Lohrey; Sebastian Maneth; Mireille Bousquet-Mélou; Eric Noeth

**Main reference** M. Lohrey, S. Maneth, E. Noeth, “XML compression via DAGs,” in Proc. of the 16th Int’l Conf. on Database Theory (ICDT’13), pp. 69–80, ACM, 2013.

**URL** <http://dx.doi.org/10.1145/2448496.2448506>

For XML this achieves high compression ratios due to their repetitive mark up. Unranked trees are often represented through first child/next sibling (fcns) encoded binary trees. We study the difference in size (i.e., number of edges) of minimal dag versus minimal dag of the fcns encoded binary tree. One main finding is that the size of the dag of the binary tree can never be smaller than the square root of the size of the minimal dag, and that there are examples that match this bound. We introduce a new combined structure, the “hybrid dag”, which is guaranteed to be smaller than (or equal in size to) both dags. Interestingly, we find through experiments that last child/previous sibling encodings are much better for XML compression via dags, than fcns encodings. This is because optional elements are more likely to appear towards the end of child sequences. The talk is based on an ICDT’2013 paper with title “XML Compression via DAGs” coauthored with Eric Noeth and Markus Lohrey.

At the end of the talk we present some new results about the expected sizes of unranked and binary DAGs. The new results can be found in the long version of the above ICDT paper, authored by Lohrey, Maneth, Noeth, and Mireille Bousquet-Mélou.

### 3.12 Succinct Data Structures

*J. Ian Munro (University of Waterloo, CA)*

**License** © Creative Commons BY 3.0 Unported license  
© J. Ian Munro

**Main reference** J.I. Munro, S.S. Rao, “Succinct Representations of Data Structures,” Chapter 37 in Mehta and Sahni (eds.), Handbook of Data Structures and Applications, Chapman & Hall/CRC, 2005.

In this talk we give an overview of the historical development of succinct data structures for the representation of graphs from the late 1980’s to the present. Early work focused on trees and planar graphs in space roughly the information theoretic minimum while supporting an increasing array of navigation operations in constant time. A variety of tree representation protocols was developed to support these operations. Much, perhaps even most, of the work was motivated by applications to text indexing. Later work on trees led to some unification of the representation protocols. Other work led to succinct representations of combinatorial structures such as groups, functions, permutations and partial orders. These ideas fed back into applications to tree representations and graph representations in general. Other aspects dealt with lower bounds and time space tradeoffs.

### 3.13 Indexing Highly Repetitive Collections

*Gonzalo Navarro (University of Chile, CL)*

**License** © Creative Commons BY 3.0 Unported license  
© Gonzalo Navarro

**Main reference** G. Navarro, “Indexing Highly Repetitive Collections,” in Proc. of the 23rd Int’l Workshop on Combinatorial Algorithms (IWOCA’12), LNCS, Vol. 7643, pp. 274–279, Springer, 2012.

**URL** [http://dx.doi.org/10.1007/978-3-642-35926-2\\_29](http://dx.doi.org/10.1007/978-3-642-35926-2_29)

The need to index and search huge highly repetitive sequence collections is rapidly arising in various fields, including computational biology, software repositories, versioned collections, and others. In this talk we describe the progress made along three research lines to address the problem: compressed suffix arrays, grammar compressed indexes, and Lempel-Ziv compressed indexes. Those lines offer progressively better compression but less search efficiency, which raises the challenge of achieving the best in both aspects. Other extended problems, such as searching in a range of versions, document listing, searching for complex patterns, etc. are outlined at the end.

### 3.14 Categorical Range Reporting

*Yakov Nekrich (University of Kansas, US)*

**License** © Creative Commons BY 3.0 Unported license  
© Yakov Nekrich

In colored range reporting problem a set of colored points is stored in a data structure. For a query rectangle  $Q$ , we must enumerate all distinct colors of points in  $Q$ . In this talk we give an overview of previous and new results and techniques for this important problem.

### 3.15 How to Cook a Poset

*Patrick K. Nicholson (University of Waterloo, CA)*

**License** © Creative Commons BY 3.0 Unported license  
© Patrick K. Nicholson

**Joint work of** Munro, J. Ian; Nicholson, Patrick K.

**Main reference** J. Ian Munro, P.K. Nicholson, “Succinct Posets,” in Proc. of the 20th Annual European Symp. on Algorithms (ESA’12), LNCS, Vol. 7501, pp. 743–754, Springer, 2012.

**URL** [http://dx.doi.org/10.1007/978-3-642-33090-2\\_64](http://dx.doi.org/10.1007/978-3-642-33090-2_64)

In this talk we survey data structures for representing partially ordered sets, or posets. The first part of the talk provides definitions of terminology related to posets. The second part is a survey of data structure results. Our main focus are the recent results of Farzan and Fischer (ISAAC 2011), and Munro and Nicholson (ESA 2012), which apply techniques from the area of succinct data structures.

### 3.16 Bioinformatics Algorithms: Sequence Analysis, Genome Rearrangements, and Phylogenetic Reconstruction

*Enno Ohlebusch (Universität Ulm, DE)*

**License** © Creative Commons BY 3.0 Unported license  
© Enno Ohlebusch

Powerful new techniques have revolutionized the field of molecular biology. The vast amount of DNA sequence information produced by next-generation sequencers demands new bioinformatics algorithms to analyze the data.

This book provides an introduction to algorithms and data structures that operate efficiently on strings (especially those used to represent long DNA sequences). It focuses on algorithms for sequence analysis (string algorithms), but also covers genome rearrangement problems and phylogenetic reconstruction methods.

### 3.17 Wavelet Tries

*Giuseppe Ottaviano (University of Pisa, IT)*

**License** © Creative Commons BY 3.0 Unported license  
© Giuseppe Ottaviano

**Joint work of** Grossi, Roberto; Ottaviano, Giuseppe

**Main reference** R. Grossi, G. Ottaviano, “The Wavelet Trie: Maintaining an Indexed Sequence of Strings in Compressed Space,” arXiv:1204.3581v1 [cs.DS], 2012.

**URL** <http://arxiv.org/abs/1204.3581v1>

We introduce and study the problem of compressed indexed sequence of strings, i.e. representing indexed sequences of strings in nearly-optimal compressed space, while preserving provably good performance for the supported operations. We present a new data structure for this problem, the Wavelet Trie, which combines the classical Patricia trie with the wavelet tree, a succinct data structure for storing compressed sequences. The resulting Wavelet Trie smoothly adapts to a sequence of strings that changes over time. It improves on the state-of-the-art compressed data structures by supporting a dynamic alphabet (i.e., the set of distinct strings) and prefix queries, both crucial requirements in the aforementioned applications, and on traditional indexes by reducing space occupancy to close to the entropy of the sequence.

### 3.18 Lightweight Lempel-Ziv Parsing

*Simon J. Puglisi (University of Helsinki, FI)*

**License** © Creative Commons BY 3.0 Unported license  
© Simon J. Puglisi

**Joint work of** Puglisi, Simon J.; Kempa, Dominik; Kärkkäinen, Juha

**Main reference** J. Kärkkäinen, D. Kempa, S. J. Puglisi, “Lightweight Lempel-Ziv Parsing,” in Proc. of the 12th Int’l Symp. on Experimental Algorithms (SEA’13), LNCS, Vol. 7933, pp. 139–150, Springer, 2013.

**URL** [http://dx.doi.org/10.1007/978-3-642-38527-8\\_14](http://dx.doi.org/10.1007/978-3-642-38527-8_14)

We introduce a new approach to LZ77 factorization that uses  $O(n/d)$  words of working space and  $O(dn)$  time for any  $d \geq 1$  (for polylogarithmic alphabet sizes). We also describe carefully engineered implementations of alternative approaches to lightweight LZ77 factorization. Extensive experiments show that the new algorithm is superior, and particularly so at the lowest memory levels and for highly repetitive data. As a part of the algorithm, we describe new methods for computing matching statistics which may be of independent interest.

### 3.19 Encoding Top- $k$ Queries

Rajeev Raman (*University of Leicester, UK*)

**License** © Creative Commons BY 3.0 Unported license  
© Rajeev Raman

**Main reference** R. Grossi, J. Iacono, G. Navarro, R. Raman, S. Rao Ratti, “Encodings for Range Selection and Top- $k$  Queries,” in Proc. of the 21st Annual European Symp. on Algorithms (ESA’13), LNCS, Vol. 8125, pp. 553–564, Springer, 2013.

**URL** [http://dx.doi.org/10.1007/978-3-642-40450-4\\_47](http://dx.doi.org/10.1007/978-3-642-40450-4_47)

We study the problem of *encoding* the positions the top- $k$  elements of an array  $A[1..n]$  for a given parameter  $1 \leq k \leq n$ . Specifically, for any  $i$  and  $j$ , we wish create a data structure that reports the positions of the largest  $k$  elements in  $A[i..j]$  in decreasing order, *without* accessing  $A$  at query time. This is a natural extension of the well-known encoding range-maxima query problem, where only the position of the maximum in  $A[i..j]$  is sought, and finds applications in document retrieval and ranking. We give (sometimes tight) upper and lower bounds for this problem and some variants thereof for general  $k$  [1], and a solution for the specific case of  $k = 2$  that has better constants than the general solution above [2].

#### References

- 1 Roberto Grossi, John Iacono, Gonzalo Navarro, Rajeev Raman and S. Srinivasa Rao. *Encodings for Range Selection and Top- $k$  Queries*. In Proc. ESA, LNCS 8125, pp. 553-564, Springer, 2013.
- 2 Pooya Davoodi, Gonzalo Navarro, Rajeev Raman and S. Srinivasa Rao. *Encoding Range Minimum Queries*. Manuscript, 2013.

### 3.20 Compressed Pattern Matching on Terms

Manfred Schmidt-Schauss (*Goethe-Universität Frankfurt am Main, DE*)

**License** © Creative Commons BY 3.0 Unported license  
© Manfred Schmidt-Schauss

Terms that are compressed with a singleton tree grammar are considered. The fully compressed pattern (sub-)match within a compressed term is analysed, trying to find the special cases which permit a polynomial time algorithm. Such cases are: (i) where compressed patterns contain every variable at most once; (ii) the pattern is DAG-compressed. It is open whether there exists a polynomial time algorithm for the general case.

### 3.21 Top- $k$ Document Retrieval

Rahul Shah (*Louisiana State University, US*)

**License** © Creative Commons BY 3.0 Unported license  
© Rahul Shah

**Joint work of** Hon, Wing-Kai; Shah, Rahul; Thankachan, Sharma T.; Vitter, Jeffrey S.

Document retrieval is a special type of pattern matching that is closely related to information retrieval and web searching. In this problem, the data consist of a collection of text documents, and given a query pattern  $P$ , we are required to report all the documents (not all the occurrences) in which this pattern occurs. In addition, the notion of *relevance* is commonly applied to rank all the documents that satisfy the query, and only those documents

with the highest relevance are returned. Such a concept of relevance has been central in the effectiveness and usability of present day search engines like Google, Bing, Yahoo, or Ask. When relevance is considered, the query has an additional input parameter  $k$ , and the task is to report only the  $k$  documents with the highest relevance to  $P$ , instead of finding all the documents that contain  $P$ . For example, one such relevance function could be the frequency of the query pattern in the document. In the information retrieval literature, this task is best achieved by using inverted indexes. However, if the query consists of an arbitrary string—which can be a partial word, multiword phrase, or more generally any sequence of characters—we cannot take advantages of the word boundaries and we need a different approach. This leads to one of the active research topics in string matching and text indexing community in recent years, and various aspects of the problem have been studied, such as space-time tradeoffs, practical solutions, multipattern queries, and I/O-efficiency. In this talk, we review some of the initial frameworks for designing such indexes and also summarize more recent developments in this area.

### 3.22 Semi-local LCS: Superglue for string comparison

*Alexander Tiskin (University of Warwick, UK)*

License © Creative Commons BY 3.0 Unported license  
© Alexander Tiskin

The computation of a longest common subsequence (LCS) between two strings is a classical algorithmic problem. A generalisation of this problem, which we call semi-local LCS, asks for the LCS between a string and all substrings of another string, and/or the LCS between all prefixes of one string and all suffixes of another. This generalised problem turns out to be fundamental whenever a solution to a string comparison or approximate matching problem has to be “glued together” from its solutions on substrings: for example, approximate matching in a compressed string; comparing strings in parallel; dynamic support of a string comparison score. The semi-local LCS problem has an elegant algebraic structure, expressed by the monoid of “seaweed braids” (i.e., the 0-Hecke monoid of the symmetric group). It also has surprising connections with computational geometry, planar graph algorithms, comparison networks, as well as practical applications in computational molecular biology. We discuss efficient algorithms for the semi-local LCS problem, and survey some related results and applications.

### 3.23 Distributed String Mining

*Niko Vaelimaeki (University of Helsinki, FI)*

License © Creative Commons BY 3.0 Unported license  
© Niko Vaelimaeki

**Joint work of** Vaelimaeki, Niko; Puglisi, Simon J.

**Main reference** N. Välimäki, S.J. Puglisi, “Distributed String Mining for High-Throughput Sequencing Data,” in Proc. of the 12th Int’l Workshop on Algorithms in Bioinformatics (WABI’12), LNCS, Vol. 7534, pp. 441–452, Springer, 2012.

**URL** [http://dx.doi.org/10.1007/978-3-642-33122-0\\_35](http://dx.doi.org/10.1007/978-3-642-33122-0_35)

The goal of frequency constrained string mining is to extract substrings that discriminate two (or more) datasets. Known solutions to the problem range from an optimal time algorithm

to different time-space tradeoffs. However, all of the existing algorithms have been designed to be run in a sequential manner and require that the whole input fits the main memory. Due to these limitations, the existing algorithms are practical only up to a few gigabytes of input. We introduce a distributed algorithm that has a novel time-space tradeoff and, in practice, achieves a significant reduction in both memory and time compared to state-of-the-art methods. To demonstrate the feasibility of the new algorithm, our study includes comprehensive tests on large-scale metagenomics data. We also study the cost of renting the required infrastructure from, e.g. Amazon EC2. Our distributed algorithm is shown to be practical on terabyte-scale inputs and affordable on rented infrastructure.

#### References

- 1 Nieves R. Brisaboa, Rodrigo Canovas, Francisco Claude, Miguel A. Martinez-Prieto and Gonzalo Navarro. *Compressed String Dictionaries*. In: Proc. SEA, LNCS 6630, pp. 136–147. Springer, 2011.
- 2 Roberto Grossi and Giuseppe Ottaviano. *Fast Compressed Tries through Path Decompositions*. In: Proc. ALENEX, pp. 65–74, SIAM, 2012.

## Participants

- Djamal Belazzougui  
University of Helsinki, FI
- Philip Bille  
Technical University of Denmark  
– Lyngby, DK
- Stefan Böttcher  
Universität Paderborn, DE
- Francisco Claude  
University of Waterloo, CA
- Henning Fernau  
Universität Trier, DE
- Johannes Fischer  
KIT – Karlsruhe Institute of  
Technology, DE
- Travis Gagie  
University of Helsinki, FI
- Paweł Gawrychowski  
MPI für Informatik –  
Saarbrücken, DE
- Roberto Grossi  
University of Pisa, IT
- Shunsuke Inenaga  
Kyushu University, JP
- Artur Jeż  
MPI für Informatik –  
Saarbrücken, DE
- Juha Kärkkäinen  
University of Helsinki, FI
- Susana Ladra Gonzalez  
University of La Coruna, ES
- Alejandro Lopez-Ortiz  
University of Waterloo, CA
- Veli Mäkinen  
University of Helsinki, FI
- Sebastian Maneth  
University of Oxford, GB
- J. Ian Munro  
University of Waterloo, CA
- Gonzalo Navarro  
University of Chile, CL
- Yakov Nekrich  
Univ. of Kansas – Lawrence, US
- Patrick K. Nicholson  
University of Waterloo, CA
- Enno Ohlebusch  
Universität Ulm, DE
- Giuseppe Ottaviano  
University of Pisa, IT
- Simon J. Puglisi  
University of Helsinki, FI
- Rajeev Raman  
University of Leicester, GB
- Manfred Schmidt-Schauss  
Goethe-Universität Frankfurt am  
Main, DE
- Diego Seco  
University of Concepcion, CL
- Rahul Shah  
Louisiana State University, US
- Yasuo Tabei  
Hokkaido University, JP
- Sharma V. Thankachan  
Louisiana State University, US
- Alexander Tiskin  
University of Warwick, GB
- Koji Tsuda  
CBRC – Tokyo, JP
- Niko Välimäki  
University of Helsinki, FI
- Rossano Venturini  
University of Pisa, IT
- Oren Weimann  
Haifa University, IL

