

Data Structures for the Cloud and External Memory Data

Edited by

Gerth Stølting Brodal¹, Ulrich Carsten Meyer², Markus E. Nebel³,
and Robert Sedgewick⁴

1 Aarhus University, DK, gerth@cs.au.dk

2 Goethe-Universität – Frankfurt a. M., DE, umeyer@cs.uni-frankfurt.de

3 Universität Bielefeld, DE, nebel@techfak.uni-bielefeld.de

4 Princeton University, US, rs@cs.princeton.edu

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 16101 “Data Structures for the Cloud and External Memory Data”. In today’s computing environment vast amounts of data are processed, exchanged and analyzed. The manner in which information is stored profoundly influences the efficiency of these operations over the data. In spite of the maturity of the field many data structuring problems are still open, while new ones arise due to technological advances. The seminar covered both recent advances in the “classical” data structuring topics as well as new models of computation adapted to modern architectures, scientific studies that reveal the need for such models, applications where large data sets play a central role, modern computing platforms for very large data, and new data structures for large data in modern architectures. The extended abstracts included in this report contain both recent state of the art advances and lay the foundation for new directions within data structures research.

Seminar January 27 – February 1, 2019 – <http://www.dagstuhl.de/19051>

2012 ACM Subject Classification Theory of computation → Data structures design and analysis, Theory of computation → Design and analysis of algorithms

Keywords and phrases algorithms, big data, cloud computing, data structures, external memory methods, large data sets, web-scale

Digital Object Identifier 10.4230/DagRep.9.1.104

Edited in cooperation with Manuel Penschuck


1 Executive Summary

Gerth Stølting Brodal (Aarhus University, DK)

Ulrich Carsten Meyer (Goethe-Universität – Frankfurt a. M., DE)

Markus E. Nebel (TU Kaiserslautern, DE)

Robert Sedgewick (Princeton University, US)

License  Creative Commons BY 3.0 Unported license

© Gerth Stølting Brodal, Ulrich Carsten Meyer, Markus E. Nebel, and Robert Sedgewick

About the Seminar

Data structures provide ways of storing and manipulating data and information that are appropriate for the computational model at hand. Every such model relies on assumptions that we have to keep questioning. The aim of this seminar was to exchange ideas for new algorithms and data structures, and to discuss our models of computations in light of recent technological advances. This Dagstuhl seminar was the 13th in a series of loosely related Dagstuhl seminars on data structures.



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

Data Structures for the Cloud and External Memory Data, *Dagstuhl Reports*, Vol. 9, Issue 1, pp. 104–124

Editors: Gerth Stølting Brodal, Ulrich Carsten Meyer, Markus E. Nebel, and Robert Sedgewick



Dagstuhl Reports

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

Topics

The presentations covered both advances in classic fields, as well as new problems and insights for recent trends in computing. In particular, Johnson (Section 3.12) and Muth (Section 3.17) reported on models and research opportunities in the cloud and external memory motivated by practical demands.

A number of talks highlighted technical challenges in storing and processing large datasets: Bast (Section 3.2) demonstrated the knowledge database QLever and discussed algorithmic aspects. Distributed frameworks were presented by Bingmann (Section 3.4) reporting on the progress of Thrill while focusing on parallel external sorting and by Yan (Section 3.32) who introduced G-thinker. Farach-Colton (Section 3.7) analyzed the slow-down of various filesystems caused by updates over time. Owens (Section 3.19) discusses intricacies of GPUs and presented efficient and practical data structures for this hardware.

In order to mitigate the impact of huge datasets, streaming and online algorithms were considered. Martinez (Section 3.15) discussed Affirmative Sampling which takes uniform samples of a stream and adapts the sample size to the stream's diversity. Sedgewick (Section 3.26) revisited the cardinality estimation problem and proposed the HyperBitBit algorithm. A matching of requests to resources in an online setting was covered by Raghvendra (Section 3.22). Similarly, Mehlhorn (Section 3.16) presented a solution to assigning indivisible resources approximately optimizing the social welfare.

Nebel (Section 3.18) and Wild (Section 3.31) proposed and analyzed tree-based data structures. Additionally, various aspects on more general graph processing were covered ranging from their enumeration (Lumbroso, Section 3.14) and random sampling (Penschuck, Section 3.20), over representations for k -connectivity (Pettie, Section 3.21) to the detection of substructures (Silvestri, Section 3.28 and Tarjan, Section 3.29).

Regarding the complexity of graph algorithms, Fagerberg (Section 3.23) presented new lower bounds on the reorganisation cost of B -trees, while Thankachan (Section 3.30) gave hardness results on the recognizability of Wheeler graphs. Kopelowitz (Section 3.13) considered the complexity of data structures for the set-disjointness problem. Emphasizing cloud-related security concerns, Jacob (Section 3.11) showed that a range of simple data structures have to incur an $\Omega(\log n)$ overhead if one wants to prevent information leakage via their access patterns.

Problems involving large text corpora were considered by Fischer (Section 3.8) presenting an external memory bi-directional compression scheme, by Golin (Section 3.9) discussing AIFV codes, and by Salinger (Section 3.24) analyzing persistent full-text indices for versioned documents.

Data structures using hashing were examined by Conway (Section 3.5), Dietzfelbinger (Section 3.6), Even and Sanders (Section 3.25). Bender (Section 3.3) discussed variants of Bloom filters which adapt based on past queries.

Afshani (Section 3.1) presented Fragile Complexity, a novel model of computation with an element-centric cost function, and gave bounds for various classical problems. Iacono (Section 3.10) proposed to model locality-of-reference more explicitly and compared his proposal to the external memory and cache-oblivious model. Sen (Section 3.27) proposed the novel paradigm HAIbrid augmenting classic data structures with artificial intelligence.

Final Thoughts

The organizers would like to thank the Dagstuhl team for their continuous support; the welcoming atmosphere made the seminar both highly productive and enjoyable. They also thank all participants for their contributions to this seminar.

2 Table of Contents

Executive Summary

Gerth Stölting Brodal, Ulrich Carsten Meyer, Markus E. Nebel, and Robert Sedgewick 104

Overview of Talks

Fragile Complexity of Comparison-based Algorithms <i>Peyman Afshani</i>	108
SPARQL Autocompletion – a Quick Intro and a Nice Open Problem <i>Hannah Bast</i>	108
Bloom Filters, Adaptivity, and the Dictionary Problem <i>Michael A. Bender</i>	109
Thrill: Pipelined External Memory Processing in the Cloud with C++ <i>Timo Bingmann</i>	109
Optimal Hashing in External Memory <i>Alexander Conway</i>	110
Constant-Time Retrieval with $O(\log m)$ Extra Bits <i>Martin Dietzfelbinger</i>	110
Dictionary Fragmentation, Affine IOs & File System Aging <i>Martin Farach-Colton</i>	111
Bidirectional Text Compression in External Memory <i>Johannes Fischer</i>	111
Polynomial Time Algorithms for Constructing Optimal AIFV Codes <i>Mordecai Golin</i>	112
The cache-oblivious model is better than you thought <i>John Iacono</i>	112
Lower Bounds for Oblivious Data Structures <i>Riko Jacob</i>	113
Modeling (and Other Research) Opportunities in Cloud and External Memory <i>Rob Johnson</i>	113
The Optimal (?) Complexity of Set-disjointness Data-structures <i>Tsvi Kopelowitz</i>	114
Enumerations Derived from Compact Encodings <i>Jérémie Lumbroso</i>	114
Affirmative Sampling <i>Conrado Martinez</i>	115
Fair Division of Indivisible Goods <i>Kurt Mehlhorn</i>	115
Reflections On Cost <i>Robert Muth</i>	116
Median-of- k Jumlists and Dangling-Min BSTs <i>Markus E. Nebel and Sebastian Wild</i>	116

What We Learned About Dynamic GPU Data Structures <i>John D. Owens</i>	117
Parallel and I/O-efficient Randomisation of Massive Networks using Global Curve- ball Trades <i>Manuel Penschuck and Ulrich Carsten Meyer</i>	117
A Cactus-type Structure for Vertex Connectivity? <i>Seth Pettie</i>	118
An Algorithm for Real-Time Matching of Requests to Resources <i>Sharath Raghvendra</i>	118
What is the Cost of Staying in Perfect Shape (if You are a B-Tree)? <i>Rolf Fagerberg and Ulrich Carsten Meyer</i>	119
Persistent Data Structures for Full-Text Indexes <i>Alejandro Salinger</i>	119
Hashing with Linear Probing and Referential Integrity <i>Peter Sanders</i>	120
Cardinality Estimation: A Poster Child for Algorithm Science <i>Robert Sedgewick</i>	120
HAIbrid Algorithms <i>Siddhartha Sen</i>	120
External Memory Output Sensitive Triangle Enumeration <i>Francesco Silvestri</i>	121
Concurrent Connected Components <i>Robert Endre Tarjan</i>	121
On the Hardness and Inapproximability of Recognizing Wheeler Graphs <i>Sharma V. Thankachan</i>	121
Entropy Trees & Range-Minimum Queries in Optimal Average-Case Space <i>Sebastian Wild</i>	122
Mining Subgraphs from a Big Graph: Solution and Challenges <i>Da Yan</i>	122
Participants	124

3 Overview of Talks

3.1 Fragile Complexity of Comparison-based Algorithms

Peyman Afshani (Aarhus University, DK)

License  Creative Commons BY 3.0 Unported license
 Peyman Afshani

Joint work of Peyman Afshani, Rolf Fagerberg, David Hammer, Riko Jacob, Irina Kostitsyna, Ulrich Meyer, Manuel Penschuck, Nodari Sitchinava

Main reference Peyman Afshani, Rolf Fagerberg, David Hammer, Riko Jacob, Irina Kostitsyna, Ulrich Meyer, Manuel Penschuck, Nodari Sitchinava: “Fragile Complexity of Comparison-Based Algorithms”, CoRR, Vol. abs/1901.02857, 2019.

URL <https://arxiv.org/abs/1901.02857>

We initiate a study of algorithms with a focus on the computational complexity of individual elements, and introduce the fragile complexity of comparison-based algorithms as the maximal number of comparisons any individual element takes part in. We give a number of upper and lower bounds on the fragile complexity for fundamental problems, including Minimum, Selection, Sorting and Heap Construction. The results include both deterministic and randomized upper and lower bounds, and demonstrate a separation between the two settings for a number of problems. The depth of a comparator network is a straight-forward upper bound on the worst case fragile complexity of the corresponding fragile algorithm. We prove that fragile complexity is a different and strictly easier property than the depth of comparator networks, in the sense that for some problems a fragile complexity equal to the best network depth can be achieved with less total work and that with randomization, even a lower fragile complexity is possible.

3.2 SPARQL Autocompletion – a Quick Intro and a Nice Open Problem

Hannah Bast (Universität Freiburg, DE)

License  Creative Commons BY 3.0 Unported license
 Hannah Bast

I gave a brief introduction to knowledge bases and Wikidata, the largest general-purpose knowledge base at the time of this talk (over 7 billion triples). I showed some demos of QLever, our query engine that supports SPARQL queries on knowledge bases as large and complex as Wikidata (SPARQL is the de facto standard query language for knowledge bases). In particular, I showed QLever’s nifty autocompletion feature and I formulated an interesting open problem that arises in the context of making this autocompletion more efficient for very large knowledge bases like Wikidata. The problem in a nutshell: given a set S of sets, compute a set P of “patterns” (frequent subsets of sets from S) such that sets from S can be written as the disjoint union of few such patterns. See the slides of the talk for details.

3.3 Bloom Filters, Adaptivity, and the Dictionary Problem

Michael A. Bender (Stony Brook University, US)

License © Creative Commons BY 3.0 Unported license
© Michael A. Bender

Joint work of Michael A. Bender, Martin Farach-Colton, Mayank Goswami, Rob Johnson, Sam McCauley, Shikha Singh

Main reference Michael A. Bender, Martin Farach-Colton, Mayank Goswami, Rob Johnson, Samuel McCauley, Shikha Singh: “Bloom Filters, Adaptivity, and the Dictionary Problem”, in Proc. of the 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018, pp. 182–193, IEEE Computer Society, 2018.

URL <https://doi.org/10.1109/FOCS.2018.00026>

A Bloom filter (or alternative) maintains a compact, probabilistic representation of a set S of keys from a universe U . The price of being small is that there is a (bounded) probability of false positives.

This talk presets alternatives to Bloom filters that are faster, more space efficient, and support a wider range of operations. We show how these filters can adapt based on the results of past queries.

Joint work with Martin Farach-Colton, Mayank Goswami, Rob Johnson, Sam McCauley, and Shikha Singh.

3.4 Thrill: Pipelined External Memory Processing in the Cloud with C++

Timo Bingmann (KIT – Karlsruhe Institut für Technologie, DE)

License © Creative Commons BY 3.0 Unported license
© Timo Bingmann

Joint work of Timo Bingmann, Simon Gog, Florian Kurpicz

Main reference Timo Bingmann, Simon Gog, Florian Kurpicz: “Scalable Construction of Text Indexes with Thrill”, in Proc. of the IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018, pp. 634–643, IEEE, 2018.

URL <https://doi.org/10.1109/BigData.2018.8622171>

We present on-going work on our distributed external Big Data processing framework Thrill [1]. It is a C++ framework consisting of a set of basic scalable algorithmic primitives like mapping, reducing and sorting, which can be combined into larger more complex algorithms, such as WordCount, PageRank, k -means clustering, and suffix sorting. These complex algorithms can then be run on very large inputs using a distributed computing cluster with external memory on each host.

We discuss how external memory is used in Thrill and the abstractions with which pipelined data processing loops are built. In this context we present current work on accelerating the distributed sorting operation with online splitter selection. As a case study of Thrill and its current external memory sorting implementation we present our freshly published results on distributed suffix array construction algorithms run on the Amazon EC2 Cloud.

References

- 1 Bingmann, Timo, et al. “Thrill: High-performance algorithmic distributed batch data processing with C++.” 2016 IEEE International Conference on Big Data (Big Data).

3.5 Optimal Hashing in External Memory

Alexander Conway (Rutgers University – Piscataway, US)

License  Creative Commons BY 3.0 Unported license
 © Alexander Conway

Hash tables are a ubiquitous class of dictionary data structures. However, standard hash table implementations do not translate well into the external memory model, because they do not incorporate locality for insertions. Iacono and Patrascu established an update/query tradeoff curve for external hash tables: a hash table that performs insertions in $\mathcal{O}(\lambda/B)$ amortized IOs require $\Omega(\log_\lambda N)$ expected IOs for queries, where N is the number of items that can be stored in the data structure, B is the size of a memory transfer, M is the size of memory, and λ is a tuning parameter. They provide a hashing data structure that meets this curve for λ that is $\Omega(\log \log M + \log_M N)$. We present a new and simpler optimal external memory hash table, the Bundle of Arrays Hash Table (BOA). BOAs are based on size-tiered LSMs and quotient filters, and are easy to implement. The BOA is optimal for a narrower range of λ . However, the simplicity of BOAs allows them to be readily modified to achieve the following results:

1. A new external memory data structure, the Bundle of Trees Hash Table (BOT), that matches the performance of the IP hash table, while retaining some of the simplicity of the BOAs.
2. The cache-oblivious Bundle of Trees Hash Table (COBOT), the first cache-oblivious hash table. This data structure matches the optimality of BOTs and IP hash tables over the same range of λ .

3.6 Constant-Time Retrieval with $\mathcal{O}(\log m)$ Extra Bits

Martin Dietzfelbinger (TU Ilmenau, DE)

License  Creative Commons BY 3.0 Unported license
 © Martin Dietzfelbinger

Joint work of Martin Dietzfelbinger, Stefan Walzer

Main reference Martin Dietzfelbinger, Stefan Walzer: “Constant-Time Retrieval with $\mathcal{O}(\log m)$ Extra Bits”, in Proc. of the 36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany, LIPIcs, Vol. 126, pp. 24:1–24:16, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2019.

URL <http://dx.doi.org/10.4230/LIPIcs.STACS.2019.24>

For a set U (the universe), retrieval is the following problem. Given a finite subset S of U of size m and $f: S \rightarrow \{0, 1\}^r$ for a small constant r , build a data structure D_f with the property that for a suitable query algorithm we have $\text{query}(D_f, x) = f(x)$ for all $x \in S$. For $x \in U \setminus S$ the value $\text{query}(D_f, x)$ is arbitrary in $\{0, 1\}^r$. The number of bits needed for D_f should be $(1 + \varepsilon)rm$ with overhead $\varepsilon = \varepsilon(m) \geq 0$ as small as possible, while the query time should be small. Of course, the time for constructing D_f is relevant as well. We assume fully random hash functions on U with constant evaluation time are available. It is known that with ε about 0.09 one can achieve linear construction time and constant query time, and with overhead $\varepsilon_k \sim e^{-k}$ it is possible to have $\mathcal{O}(k)$ query time and $\mathcal{O}(m^{1+\alpha})$ construction time, for arbitrary $\alpha > 0$. Furthermore, a theoretical construction with $\varepsilon = \mathcal{O}((\log \log m)/\sqrt{\log m})$ gives constant query time and linear construction time. Known constructions avoiding all overhead, except for a seed value of size $\mathcal{O}(\log \log m)$, require logarithmic query time. In this paper, we present a method for treating the retrieval problem with overhead $\varepsilon = \mathcal{O}((\log m)/m)$,

which corresponds to $\mathcal{O}(1)$ extra memory words ($\mathcal{O}(\log m)$ bits), and an extremely simple, constant-time query operation. The price to pay is a construction time of $\mathcal{O}(m^2)$. We employ the usual framework for retrieval data structures, where construction is effected by solving a sparse linear system of equations over the 2-element field F_2 and a query is effected by a dot product calculation. Our main technical contribution is the design and analysis of a new and natural family of sparse random linear systems with m equations and $(1 + \varepsilon)m$ variables, which combines good locality properties with high probability of having full rank. Paying a larger overhead of $\varepsilon = \mathcal{O}((\log m)/m^\alpha)$, the construction time can be reduced to $\mathcal{O}(m^{1+\alpha})$ for arbitrary constant $0 < \alpha < 1$. In combination with an adaptation of known techniques for solving sparse linear systems of equations, our approach leads to a highly practical algorithm for retrieval. In a particular benchmark with $m = 10^7$ we achieve an order-of-magnitude improvement over previous techniques with ε about 0.24% instead of the previously best result of ε about 3%, with better query time and no significant sacrifices in construction time.

3.7 Dictionary Fragmentation, Affine IOs & File System Aging

Martin Farach-Colton (Rutgers University – Piscataway, US)

License © Creative Commons BY 3.0 Unported license
© Martin Farach-Colton

B-trees fragment because nodes are small. Be-trees don't fragment because nodes are large. We test this hypothesis on file systems and find the modern file systems age by factors of 20 or greater, except for BetrFS, which does not age at all.

3.8 Bidirectional Text Compression in External Memory

Johannes Fischer (TU Dortmund, DE)

License © Creative Commons BY 3.0 Unported license
© Johannes Fischer

Joint work of Johannes Fischer, Patrick Dinklage, Jonas Ellert, Dominik Köppl, Manuel Penschuk

We present a new algorithm for text compression in external memory. It works by substituting repeated substrings by references, which can point either to the left or to the right of the current position (unlike Lempel-Ziv-77, which can only refer to the left). The algorithm is based on the “permuted longest common prefix array”, which can be computed efficiently while suffix-sorting the text, for which good external memory algorithms exist. We test our algorithms on inputs of up to 128 GiB on a computer with 16 GiB of RAM, where it uses much less space than the best external memory implementations of the LZ7-algorithm, while showing a similar compression rate.

3.9 Polynomial Time Algorithms for Constructing Optimal AIFV Codes

Mordecai Golin (HKUST – Kowloon, HK)

License © Creative Commons BY 3.0 Unported license
© Mordecai Golin

Joint work of Mordecai Golin, Elfarouk Harb

Main reference Mordecai J. Golin, Elfarouk Y. Harb: “Polynomial Time Algorithms for Constructing Optimal AIFV Codes”, in Proc. of the Data Compression Conference, DCC 2019, Snowbird, UT, USA, March 26-29, 2019, pp. 231–240, IEEE, 2019.

URL <http://dx.doi.org/10.1109/DCC.2019.00031>

Huffman Codes are *optimal* Fixed-to-Variable (FV) codes if every source symbol can only be encoded by one codeword. Relaxing these constraints permits constructing better FV codes. More specifically, recent work has shown that AIFV codes can beat Huffman coding. AIFV codes construct a set of different coding trees between which the code alternates and are only *almost instantaneous* (AI). This means that decoding a word might require a delay of a finite number of bits.

Current algorithms for constructing optimal AIFV codes are iterative processes. One iteration step improves the current set of trees to a “better” set. The process has been proven to finitely converge to the optimal code but with but no known bounds on the convergence time.

This paper derives a geometric interpretation of the space of AIFV codes. This permits the development of new polynomially time-bounded iterative procedures for constructing optimal AIFV codes.

For the simplest case we show that a binary search procedure can replace the current iterative process. For the more complicated cases we describe how to frame the problem as a linear programming problem with an exponential number of constraints but a polynomial time separability oracle. This permits using the Grötschel, Lovász and Schrijver ellipsoid method to solve the problem in a polynomial number of steps.

This work will be presented at DCC’19.

3.10 The cache-oblivious model is better than you thought

John Iacono (UL – Brussels, BE)

License © Creative Commons BY 3.0 Unported license
© John Iacono

Joint work of John Iacono, Varunkumar Jayapaul, Ben Karsin

Main reference John Iacono, Varunkumar Jayapaul, Ben Karsin: “Locality”, CoRR, Vol. abs/1902.07928, 2019.

URL <https://arxiv.org/abs/1902.07928>

The performance of modern computation is characterized by locality of reference, that is, it is cheaper to access data that has been accessed recently than a random piece of data. This is due to many architectural features including caches, lookahead, address translation and the physical properties of a hard disk drive; attempting to model all the components that constitute the performance of a modern machine is impossible, especially for general algorithm design purposes. What if one could prove an algorithm is asymptotically optimal on all systems that reward locality of reference, no matter how it manifests itself within reasonable limits? We show that this is possible, and that algorithms that are asymptotically optimal in the cache-oblivious model are asymptotically optimal in any reasonable locality-of-reference rewarding setting. This is surprising as the cache-oblivious model envisions a particular

architectural model involving blocked memory transfer into a multi-level hierarchy of caches of varying sizes, and was not designed to directly model locality-of-reference correlated performance.

3.11 Lower Bounds for Oblivious Data Structures

Riko Jacob (IT University of Copenhagen, DK)

License © Creative Commons BY 3.0 Unported license
© Riko Jacob

Joint work of Riko Jacob, Kasper Green Larsen, Jesper Buus Nielsen

Main reference Riko Jacob, Kasper Green Larsen, Jesper Buus Nielsen: “Lower Bounds for Oblivious Data Structures”, in Proc. of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019, pp. 2439–2447, SIAM, 2019.

URL <https://doi.org/10.1137/1.9781611975482.149>

An oblivious data structure is a data structure where the memory access patterns reveals no information about the operations performed on it. Such data structures were introduced by Wang et al. [ACM SIGSAC’14] and are intended for situations where one wishes to store the data structure at an untrusted server. One way to obtain an oblivious data structure is simply to run a classic data structure on an oblivious RAM (ORAM). Until very recently, this resulted in an overhead of $\omega(\log n)$ for the most natural setting of parameters. Moreover, a recent lower bound for ORAMs by Larsen and Nielsen [CRYPTO’18] show that they always incur an overhead of at least $\Omega(\log n)$ if used in a black box manner. To circumvent the $\omega(\log n)$ overhead, researchers have instead studied classic data structure problems more directly and have obtained efficient solutions for many such problems such as stacks, queues, dequeues, priority queues and search trees. However, none of these data structures process operations faster than $\Theta(\log n)$, leaving open the question of whether even faster solutions exist. In this paper, we rule out this possibility by proving $\Omega(\log n)$ lower bounds for oblivious stacks, queues, dequeues, priority queues and search trees.

3.12 Modeling (and Other Research) Opportunities in Cloud and External Memory


Rob Johnson (VMware – Palo Alto, US)

License © Creative Commons BY 3.0 Unported license
© Rob Johnson

This talk will describe several trends in cloud computing that can serve as inspiration for new models and new algorithms and data structure research.

3.13 The Optimal (?) Complexity of Set-disjointness Data-structures

Tsvi Kopelowitz (Bar-Ilan University – Ramat Gan, IL)

License  Creative Commons BY 3.0 Unported license
© Tsvi Kopelowitz

In the set-disjointness problem the goal is to preprocess a family of sets $F = \{S_1, S_2, \dots, S_k\}$, all from universe U , so that given two sets $S_i, S_j \in F$, one can quickly establish whether the two sets are disjoint or not. If N is the sum of the sizes of the sets in F , then let N^p be the preprocessing time and let N^q be the query time. A folklore combinatorial algorithm has a tradeoff curve of $p + q = 2$, which is optimal (up to sub-polynomial terms) for combinatorial algorithms, assuming the combinatorial BMM conjecture. In SODA'16, Kopelowitz, Pettie, and Porat showed that, based on the 3SUM, there is a conditional lower bound curve of $p + 2q \geq 2$, and so there exists a gap between the upper bound curve and the lower bound curve when allowing non-combinatorial techniques.

In this talk we will show that both curves can be improved. Specifically, if one assumes that the constant in the exponent of fast matrix multiplication is $\omega = 2$, then one can obtain an upper bound curve of $p + 2q \geq 2$ for $q \leq 1/3$ (matching the 3SUM based lower bound for this case), and $2p + q = 3$ for $q \geq 1/3$. Moreover, we introduce a new conjecture on the time required for detecting a triangle in an unbalanced tripartite graph, which is closely related to the triangle detection conjecture for general graphs, and is used to show that the new upper bound curve for set-disjointness is tight.

3.14 Enumerations Derived from Compact Encodings

Jérémie Lumbroso (Princeton University, US)

License  Creative Commons BY 3.0 Unported license
© Jérémie Lumbroso

Joint work of Jérémie Lumbroso, Maryam Bahrani, Cédric Chauve, Éric Fusy, Jessica Shi
Main reference Jérémie O. Lumbroso, Jessica Shi: “Exponential Bounds on Graph Enumerations from Vertex Incremental Characterizations”, in Proc. of the Fifteenth Workshop on Analytic Algorithmics and Combinatorics, ANALCO 2018, New Orleans, LA, USA, January 8-9, 2018., pp. 118–132, SIAM, 2018.

URL <http://dx.doi.org/10.1137/1.9781611975062.11>

We compare several methodologies to either compute or approximate the enumeration of a combinatorial class:

1. The “compact encoding” methodology, in which an encoding of the objects of the combinatorial class is designed, and then the enumeration sequence is bounded by lower/upper-bounding the number of bits required by the encoding for any object of a given size.
2. A method using the analytic combinatorics toolset, in which a bijection allows us to reduce the objects of the combinatorial class to an easier to count class, which can then be exactly enumerated using existing techniques.
3. A “hybrid” methodology, in which we design an encoding for the combinatorial class, but then count all possible encodings using analytic combinatorics tree enumeration techniques.

We argue that the third method provides a good trade-off between technical complexity and accuracy. We illustrate these arguments on various examples taken from recent work on the enumeration of unlabeled graph classes, including papers by Nakano *et al.* [2], Lumbroso and Shi [3], and Chauve *et al.* [1].

This is joint work with several researchers: Maryam Bahrani (Princeton), Cédric Chauve (Simon Fraser University), Éric Fusy (Polytechnique), Jessica Shi (MIT).

References

- 1 Chauve, Cédric, Éric Fusy, and Jérémie Lumbroso. *An Exact Enumeration of Distance-Hereditary Graphs*, in Proceedings of the Fourteenth Meeting on Analytic Algorithmics and Combinatorics (ANALCO 2017), C. Martínez and M. D. Ward, Eds., Barcelona, Spain: Society for Industrial and Applied Mathematics, pp. 31–45, Jan. 2017. doi: 10.1137/1.9781611974775.3.
- 2 Nakano, Shin-ichi, Ryuhei Uehara, and Takeaki Uno. *A new approach to graph recognition and applications to distance-hereditary graphs*, in Journal of computer science and technology 24.3, pp. 517–533, 2009.
- 3 Lumbroso, Jérémie, and Jessica Shi.. *Exponential Bounds on Graph Enumerations from Vertex Incremental Characterizations*, in Proceedings of the Fifteenth Meeting on Analytic Algorithmics and Combinatorics (ANALCO 2018), M. Nebel and S. Wagner, Eds., New Orleans, USA: Society for Industrial and Applied Mathematics, pp. 118–132, Jan. 2018.

3.15 Affirmative Sampling

Conrado Martínez (UPC – Barcelona, ES)

License © Creative Commons BY 3.0 Unported license
© Conrado Martínez

Joint work of Conrado Martine, Jérémie Lumbroso

Affirmative Sampling is a practical and efficient novel algorithm to obtain random samples of distinct elements from a data stream, its most salient feature being that the size S of the sample will, on expectation, grow with the (unknown) number n of distinct elements in the data stream. As any distinct element has the same probability to be sampled, and the sample size is greater when the cardinality (when the “diversity”) is greater, the samples that Affirmative Sampling delivers are more representative and enable more accurate inferences than those produced by any scheme where the sample size is fixed a priori – hence, its name.

3.16 Fair Division of Indivisible Goods

Kurt Mehlhorn (MPI für Informatik – Saarbrücken, DE)

License © Creative Commons BY 3.0 Unported license
© Kurt Mehlhorn

Joint work of Bhaskar Ray Chaudhury, Yun Kuen Cheung, Jugal Garg, Naveen Garg, Martin Hoefer, Kurt Mehlhorn

Main reference Bhaskar Ray Chaudhury, Yun Kuen Cheung, Jugal Garg, Naveen Garg, Martin Hoefer, Kurt Mehlhorn: “On Fair Division for Indivisible Items”, in Proc. of the 38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, December 11–13, 2018, Ahmedabad, India, LIPIcs, Vol. 122, pp. 25:1–25:17, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2018.



URL <http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2018.25>

We consider the task of assigning indivisible goods to a set of agents in a fair manner. Our notion of fairness is Nash social welfare, i.e., the goal is to maximize the geometric mean of the utilities of the agents. Each good comes in multiple items or copies, and the utility of an agent diminishes as it receives more items of the same good. The utility of a bundle of items for an agent is the sum of the utilities of the items in the bundle. Each agent has

a utility cap beyond which he does not value additional items. We give a polynomial time approximation algorithm that maximizes Nash social welfare up to a factor of $e^{1/e} \approx 1.445$. The computed allocation is Pareto-optimal and approximates envy-freeness up to one item up to a factor of $2 + \varepsilon$.

3.17 Reflections On Cost

Robert Muth (Google – New York, US)



License  Creative Commons BY 3.0 Unported license
 Robert Muth

This informal talk is a practitioner's view of running algorithms in the cloud at Google.

We discuss various forms of cost metrics, including total cost of ownership (TCO). We point out how cloud needs differ from the classical space/time metrics, make some predictions about how certain cost models will change and point out opportunities for new research and how theory can inform practice.

3.18 Median-of- k Jumplists and Dangling-Min BSTs

Markus E. Nebel (Universität Bielefeld, DE) and Sebastian Wild (University of Waterloo, CA)

License  Creative Commons BY 3.0 Unported license
 Markus E. Nebel and Sebastian Wild
Joint work of Markus E. Nebel, Elisabeth Neumann, Sebastian Wild
Main reference Markus E. Nebel, Elisabeth Neumann, Sebastian Wild: “Median-of- k Jumplists”, CoRR, Vol. abs/1609.08513, 2016.
URL <http://arxiv.org/abs/1609.08513>

In this talk we discuss a variant of jumplists where the jump pointer's target is chosen as the median of a random sample of size k . We present a new search strategy called spine search which allows to skip some elements of the list which the traditional algorithm would have inspected. We prove a precise asymptotic for the average number of key comparisons performed by this new strategy which shows that larger values of k imply an improved lookup time for the jump lists tend to be more balanced. On the other hand, insertions and deletions get more costly with increasing k since the need for rebalancing gets worse. As a second improvement we analyse the effect of omitting jump-pointer for sublists of length at most w , proving that a constant fraction of pointer is saved while search costs only increase by a constant additive term.

3.19 What We Learned About Dynamic GPU Data Structures

John D. Owens (*University of California, Davis, US*)

License © Creative Commons BY 3.0 Unported license
© John D. Owens

Joint work of John Owens, Martin Farach-Colton, as well as many coauthors

In this talk I discuss four dynamic data structures we built for the GPU – log-structured merge trees, quotient filters, linked lists and hash tables built atop them, and B-trees. I discuss principles that we followed in building them and then what we learned, including lessons from mapping work to threads vs. warps, issues with contention, the use of the cache hierarchy, surprising insertion throughput results for the LSM vs. the B-tree, memory allocation, resizing, and semantics.

References

- 1 Muhammad A. Awad, Saman Ashkiani, Rob Johnson, Martín Farach-Colton, and John D. Owens. *Engineering a High-Performance GPU B-Tree*. In *Proceedings of the 24th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPOPP 2019, February 2019.
- 2 Saman Ashkiani, Martin Farach-Colton, and John D. Owens. *A Dynamic Hash Table for the GPU*. *Proceedings of the 31st IEEE International Parallel and Distributed Processing Symposium*, IPDPS 2018, pages 419–429. May 2018.
- 3 Saman Ashkiani, Shengren Li, Martin Farach-Colton, Nina Amenta, and John D. Owens. *GPU LSM: A Dynamic Dictionary Data Structure for the GPU*. In *Proceedings of the 31st IEEE International Parallel and Distributed Processing Symposium*, IPDPS 2018, pages 430–440. May 2018.
- 4 Afton Geil, Martin Farach-Colton, and John D. Owens. *Quotient Filters: Approximate Membership Queries on the GPU*. In *Proceedings of the 31st IEEE International Parallel and Distributed Processing Symposium*, IPDPS 2018, pages 451–462. May 2018.

3.20 Parallel and I/O-efficient Randomisation of Massive Networks using Global Curveball Trades

Manuel Penschuck (*Goethe-Universität – Frankfurt a. M., DE*) and Ulrich Carsten Meyer (*Goethe-Universität – Frankfurt a. M., DE*)

License © Creative Commons BY 3.0 Unported license
© Manuel Penschuck and Ulrich Carsten Meyer

Joint work of Corrie Jacobien Carstens, Michael Hamann, Ulrich Meyer, Manuel Penschuck, Hung Tran, Dorothea Wagner

Main reference Corrie Jacobien Carstens, Michael Hamann, Ulrich Meyer, Manuel Penschuck, Hung Tran, Dorothea Wagner: “Parallel and I/O-efficient Randomisation of Massive Networks using Global Curveball Trades”, in Proc. of the 26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland, LIPIcs, Vol. 112, pp. 11:1–11:15, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2018.

URL <http://dx.doi.org/10.4230/LIPIcs.ESA.2018.11>

Graph randomisation is a crucial task in the analysis and synthesis of networks. It is typically implemented as an edge switching process (ESMC) repeatedly swapping the nodes of random edge pairs while maintaining the degrees involved. We discuss EM-ES, an I/O-efficient edge switching algorithm for this setting.


In this context, Curveball is a novel approach that instead considers the whole neighbourhoods of randomly drawn node pairs. Its Markov chain converges to a uniform distribution,

and experiments suggest that it requires less steps than the established ESMC. Since trades however are more expensive, we study Curveball’s practical runtime by introducing the first efficient Curveball algorithms: the I/O-efficient EM-CB for simple undirected graphs and its internal memory pendant IM-CB.

Further, we investigate global trades processing every node in a graph during a single super step, and show that undirected global trades converge to a uniform distribution and perform superior in practice. We then discuss EM-GCB and EM-PGCB for global trades and give experimental evidence that EM-PGCB achieves the quality of the state-of-the-art ESMC algorithm EM-ES nearly one order of magnitude faster.

3.21 A Cactus-type Structure for Vertex Connectivity?

Seth Pettie (University of Michigan – Ann Arbor, US)

License  Creative Commons BY 3.0 Unported license
© Seth Pettie

It is well known that the set of all global minimum edge cuts in a graph can be succinctly represented as an $\mathcal{O}(n)$ -space “cactus tree”. In this talk I’ll survey efforts to develop an analogue of the cactus structure for k -vertex connectivity, such as the block tree ($k = 1$), the SPQR tree ($k = 2$), and the structure proposed by Cohen et al. for general k -connectivity.

3.22 An Algorithm for Real-Time Matching of Requests to Resources

Sharath Raghvendra (Virginia Polytechnic Institute – Blacksburg, US)

License  Creative Commons BY 3.0 Unported license
© Sharath Raghvendra

Main reference Sharath Raghvendra: “A Robust and Optimal Online Algorithm for Minimum Metric Bipartite Matching”, in Proc. of the Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France, LIPIcs, Vol. 60, pp. 18:1–18:16, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016.

URL <http://dx.doi.org/10.4230/LIPIcs.APPROX-RANDOM.2016.18>

Main reference Krati Nayyar, Sharath Raghvendra: “An Input Sensitive Online Algorithm for the Metric Bipartite Matching Problem”, in Proc. of the 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017, pp. 505–515, IEEE Computer Society, 2017.


URL <https://doi.org/10.1109/FOCS.2017.53>

In the online minimum-metric bipartite matching problem, we are given a set S of server locations in a metric space. The locations of requests are revealed one at a time and when a request is revealed, we must immediately and irrevocably match it to a “free” server. The cost of matching a server to request is given by the distance between the two locations (which we assume satisfies triangle inequality). The objective of this problem is to come up with a matching of servers to requests which is competitive with respect to the minimum-cost matching of S and R .

In this talk, I present an online algorithm that, in the adversarial request generation model, achieves near-optimal competitive ratio for any metric space. I will also show that this algorithm achieves an optimal competitive ratio of $2H_n - 1$ in the random arrivals model. I will show that this algorithm not only achieves near-optimal bounds in adversarial settings but also performs well in practical settings.

3.23 What is the Cost of Staying in Perfect Shape (if You are a B-Tree)?

Rolf Fagerberg (*University of Southern Denmark – Odense, DK*) and Ulrich Carsten Meyer (*Goethe-Universität – Frankfurt a. M., DE*)

License  Creative Commons BY 3.0 Unported license
© Rolf Fagerberg and Ulrich Carsten Meyer

Joint work of Rolf Fagerberg, David Hammer, Ulrich Carsten Meyer

Any B-tree has height at least $\lceil \log_B(n) \rceil$. Static B-trees achieving this are easy to build, but in the dynamic case, standard B-tree rebalancing algorithms only maintain a height within a constant factor of this optimum. We investigate exactly how close to $\lceil \log_B(n) \rceil$ the height of dynamic B-trees can be maintained and at what rebalancing costs. As usual, cost means the number of nodes accessed. We prove a lower bound on the cost of maintaining optimal height $\lceil \log_B(n) \rceil$, which shows that this cost must increase from $\Omega(1/B)$ to $\Omega(n/B)$ rebalancing per update as n grows from one power of B to the next. We also provide an almost matching upper bound, demonstrating this lower bound to be essentially tight. We then give a variant upper bound which can maintain near-optimal height at low cost. As two special cases, we can maintain optimal height for all but a vanishing fraction of values of n using $\Theta(\log_B(n))$ amortized rebalancing cost per update and we can maintain a height of optimal plus one using $O(1/B)$ amortized rebalancing cost per update. More generally, for any rebalancing budget, we can maintain (as n grows from one power of B to the next) optimal height essentially up to the point where the lower bound requires the budget to be exceeded, after which optimal height plus one is maintained. Finally, we prove that this balancing scheme gives B-trees with very good storage utilization.

3.24 Persistent Data Structures for Full-Text Indexes

Alejandro Salinger (*SAP SE – Walldorf, DE*)

License  Creative Commons BY 3.0 Unported license
© Alejandro Salinger

Joint work of Alejandro Salinger, Divya Venkatesan

We study an extension of the dynamic text collection problem in which versions of the text collection are maintained. In this problem, texts can be inserted to or deleted from any version of the collection, which in turn creates a new version of the collection. A solution should support the efficient location of a pattern in any version of the collection. An example of an application of this problem is in indexes of textual data in databases which support Multiversion Concurrency Control (MVCC). We present a full-text index which supports multi-version access while enabling users to modify any version of the indexed text collection. The index is based on the dynamic full-text indexes by Chan et al. [1] and Navarro and Mäkinen [2] and makes use of persistent data structures techniques to keep track of the versions of the collection in all data structures. We also present a simple variant of the above index which we use to compare its performance. It consists of a set of separate static indexes for the changes in the collection and achieves full persistence by maintaining a global version tree. We present an experimental evaluation of both indexes with respect to time and space usage for different operations.

This talk is based on the master's thesis of Divya Venkatesan (TU Kaiserslautern and SAP SE).

References

- 1 Chan, Ho-Leung and Hon, Wing-Kai and Lam, Tak-Wah and Sadakane, Kunihik. *Compressed Indexes for Dynamic Text Collections*. ACM Trans. Algorithms, 2007.
- 2 Mäkinen, Veli and Navarro, Gonzalo. *Dynamic Entropy-compressed Sequences and Full-text Indexes*. ACM Trans. Algorithms, 2008.

3.25 Hashing with Linear Probing and Referential Integrity

Peter Sanders (KIT – Karlsruher Institut für Technologie, DE)

License © Creative Commons BY 3.0 Unported license
© Peter Sanders

Main reference Peter Sanders: “Hashing with Linear Probing and Referential Integrity”, CoRR, Vol. abs/1808.04602, 2018.

URL <http://arxiv.org/abs/1808.04602>

We describe a variant of linear probing hash tables that never moves elements and thus supports referential integrity, i.e., pointers to elements remain valid while this element is in the hash table. This is achieved by the folklore method of marking some table entries as formerly occupied (tombstones). The innovation is that the number of tombstones is minimized. Experiments indicate that this allows an unbounded number of operations with bounded overhead compared to linear probing without tombstones (and without referential integrity).

3.26 Cardinality Estimation: A Poster Child for Algorithm Science

Robert Sedgewick (Princeton University, US)

License © Creative Commons BY 3.0 Unported license
© Robert Sedgewick

The problem of determining an approximate count of the number of items in a data stream has a rich history, dating back to a paper by Flajolet and Martin in the 1980s. This talk surveys the useful methods that have been invented, focusing on algorithms developed by Flajolet and co-authors that culminated in the HyperLogLog algorithm. These algorithms are characterized by careful analysis validated by experimentation, and are widely used today in cloud computing. The talk ends with a proposed new algorithm, HyperBitBit, that seems to provide accurate estimates for large streams using very little computation per item and only a very small amount of memory.

3.27 HAIbrid Algorithms

Siddhartha Sen (Microsoft – New York, US)

License © Creative Commons BY 3.0 Unported license
© Siddhartha Sen

HAIbrid (Human + AI) Algorithms synergize human solutions with AI to enhance the performance and adaptivity of hand-designed data structures and algorithms. These data structures and algorithms underlie our cloud storage, search, and scheduling systems. Rather

than avoiding AI or using it blindly, we seek the right combination – a new form of human-AI collaboration. I will describe a paradigm that combines reinforcement learning with the ability to ask counterfactual (“what if”) questions about any decision-making algorithm, provided there is sufficient randomness in its decisions. This paradigm can readily be applied to data structures like skip lists and treaps which are naturally randomized. Our ultimate goal is to create a “universal data structure” that delivers the best performance at every point in time, for any workload, through human + AI co-design.

3.28 External Memory Output Sensitive Triangle Enumeration

Francesco Silvestri (University of Padova, IT)

License © Creative Commons BY 3.0 Unported license
© Francesco Silvestri

Joint work of Rasmus Pagh, Francesco Silvestri

Triangle enumeration in the external memory model has been widely studied in a worst case scenario. It is not clear however if we can derive a more efficient algorithm whose I/O complexity is a function of the actual number of triangles of the input graph. In this talk, I provide a quick overview of the state-of-the-art, and describe some preliminary results on output-sensitive triangle enumeration in the external memory model.

3.29 Concurrent Connected Components

Robert Endre Tarjan (Princeton University, US)

License © Creative Commons BY 3.0 Unported license
© Robert Endre Tarjan

Joint work of Robert Endre Tarjan, Sixue (Cliff) Liu

Finding the connected components of a graph is one of the most basic graph problems. Although it is easy to find components sequentially using graph search or a disjoint set union algorithm, some important applications require finding the components of huge graphs, making sequential algorithms too slow. We describe recent progress on concurrent algorithms for this problem. Some simple algorithms seem surprisingly hard to analyze.

3.30 On the Hardness and Inapproximability of Recognizing Wheeler Graphs

Sharma V. Thankachan (University of Central Florida – Orlando, US)

License © Creative Commons BY 3.0 Unported license
© Sharma V. Thankachan

Joint work of Sharma V. Thankachan, Daniel Gibney

Main reference Daniel Gibney, Sharma V. Thankachan: “On the Hardness and Inapproximability of Recognizing Wheeler Graphs”, CoRR, Vol. abs/1902.01960, 2019.

URL <https://arxiv.org/abs/1902.01960>

In recent years the relationship between a newly defined class of graphs and several important string indexing structures has been discovered. This class of graphs, known as Wheeler graphs, were shown by Gagie et al. to model de Bruijn graphs, generalized compressed

suffix arrays, and several other BWT related structures. Moreover, the Wheeler graph axioms reveal a sufficient condition for a data structure to be indexed efficiently. In our work, we prove the NP-hardness of recognizing Wheeler graphs, in addition to providing an exponential time algorithm for the recognition problem which has better time complexity than the naive approach. We also show the APX-hardness of finding the minimum number of edges that must be removed to transform a graph into a Wheeler graph. On the other hand, we demonstrate that the dual of this optimization problem, finding the maximal Wheeler graph, admits a constant approximation.

3.31 Entropy Trees & Range-Minimum Queries in Optimal Average-Case Space

Sebastian Wild (University of Waterloo, CA)

License © Creative Commons BY 3.0 Unported license
© Sebastian Wild

Joint work of Munro, J. Ian, Nicholson, Patrick, Sebastian Wild

Main reference J. Ian Munro, Sebastian Wild: “Entropy Trees and Range-Minimum Queries In Optimal Average-Case Space”, CoRR, Vol. abs/1903.02533, 2019.

URL <https://arxiv.org/abs/1903.02533>

The range-minimum query (RMQ) problem is a fundamental data structuring task with numerous applications. Despite the fact that succinct solutions with worst-case optimal $2n + o(n)$ bits of space and constant query time are known, it has been unknown whether such a data structure can be made adaptive to the reduced entropy of random inputs (Davoodi et al. 2014). We construct a succinct data structure with the optimal $1.73n + o(n)$ bits of space on average for random RMQ instances, settling this open problem. Our solution relies on a compressed data structure for binary trees of independent interest.

References

- 1 Pooya Davoodi, Gonzalo Navarro, Rajeev Raman, and Srinivasa Rao Satti. *Encoding range minima and range top-2 queries*. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 372(2016):20130131–20130131, 2014. 10.1098/rsta.2013.0131

3.32 Mining Subgraphs from a Big Graph: Solution and Challenges

Da Yan (The University of Alabama – Birmingham, US)

License © Creative Commons BY 3.0 Unported license
© Da Yan

Joint work of Da Yan, Guimu Guo, Md Mashiur Rahman Chowdhury, M. Tamer özsu, John C.S. Lui, Weida Tan

Main reference Da Yan, Guimu Guo, Md Mashiur Rahman Chowdhury, M. Tamer Özsu, John C. S. Lui, Weida Tan: “T-thinker: a task-centric distributed framework for compute-intensive divide-and-conquer algorithms”, in Proc. of the 24th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPOPP 2019, Washington, DC, USA, February 16-20, 2019, pp. 411–412, ACM, 2019.

URL <https://doi.org/10.1145/3293883.3295709>

The problem of mining subgraphs from a big data graph finds numerous applications including social community detection and finding patterns from biological networks. Due to the high computational complexity of this problem, parallel and distributed computing is essential in order to scale to big graphs. However, existing cloud computing models such as MapReduce

and Google's Pregel are IO-bound and severely underutilize CPU cores when applied to this problem. This talk describes a new framework called G-thinker for large-scale graph mining, by briefly overviewing the parallel data structures used in G-thinker's design. G-thinker has been open-sourced at <http://www.cs.uab.edu/yanda/gthinker>. This talk will also bring some design challenges found during the development of G-thinker for open discussion.

Participants

- Peyman Afshani
Aarhus University, DK
- Hannah Bast
Universität Freiburg, DE
- Michael A. Bender
Stony Brook University, US
- Ioana Bercea
Tel Aviv University, IL
- Timo Bingmann
KIT – Karlsruher Institut für
Technologie, DE
- Gerth Stølting Brodal
Aarhus University, DK
- Alexander Conway
Rutgers University –
Piscataway, US
- Martin Dietzfelbinger
TU Ilmenau, DE
- Guy Even
Tel Aviv University, IL
- Tomer Even
Tel Aviv University, IL
- Rolf Fagerberg
University of Southern Denmark –
Odense, DK
- Martin Farach-Colton
Rutgers University –
Piscataway, US
- Johannes Fischer
TU Dortmund, DE
- Mordecai Golin
HKUST – Kowloon, HK
- Herman J. Haverkort
Universität Bonn, DE
- John Iacono
UL – Brussels, BE
- Riko Jacob
IT University of
Copenhagen, DK
- Rob Johnson
VMware – Palo Alto, US
- Tsvi Kopelowitz
Bar-Ilan University –
Ramat Gan, IL
- Moshe Lewenstein
Bar-Ilan University –
Ramat Gan, IL
- Jérémie Lumbroso
Princeton University, US
- Conrado Martinez
UPC – Barcelona, ES
- Kurt Mehlhorn
MPI für Informatik –
Saarbrücken, DE
- Ulrich Carsten Meyer
Goethe-Universität –
Frankfurt a. M., DE
- Friedhelm Meyer auf der Heide
Universität Paderborn, DE
- Ian Munro
University of Waterloo, CA
- Robert Muth
Google – New York, US
- Markus E. Nebel
Universität Bielefeld, DE
- John D. Owens
University of California at
Davis, US
- Manuel Penschuck
Goethe-Universität –
Frankfurt a. M., DE
- Seth Pettie
University of Michigan –
Ann Arbor, US
- Sharath Raghvendra
Virginia Polytechnic Institute –
Blacksburg, US
- Rajeev Raman
University of Leicester, GB
- Alejandro Salinger
SAP SE – Walldorf, DE
- Peter Sanders
KIT – Karlsruher Institut für
Technologie, DE
- Robert Sedgewick
Princeton University, US
- Siddhartha Sen
Microsoft – New York, US
- Francesco Silvestri
University of Padova, IT
- Robert Endre Tarjan
Princeton University, US
- Sharma V. Thankachan
University of Central Florida –
Orlando, US
- Sebastian Wild
University of Waterloo, CA
- Da Yan
The University of Alabama –
Birmingham, US
- Norbert Zeh
Dalhousie University –
Halifax, CA

