Report from Dagstuhl Seminar 25191

Adaptive and Scalable Data Structures

Michael A. Bender[*1], **John Iacono**[*2], **László Kozma**[*3], **Eva Rotenberg**[*4], **and Justin Dallant**[†5]

1 **Stony Brook University, US.** `bender@cs.stonybrook.edu`
2 **ULB – Brussels, BE.** `john.iacono@ulb.be`
3 **FU Berlin, DE.** `lkozma@zedat.fu-berlin.de`
4 **Technical University of Denmark – Lyngby, DK.** `eva@rotenberg.dk`
5 **ULB – Brussels, BE.** `justindallant@gmail.com`

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 25191 "Adaptive and Scalable Data Structures". Data structures govern the organization and manipulation of data in computing systems across a broad range of applications. The efficiency and scalability of data structures has profound implications, motivating continued research on the entire spectrum from theoretical to practical. As the size and complexity of data sets increases and as the underlying computing infrastructure changes, data structures need to be continually redesigned with scalability in mind. Classical data structures also need reevaluation to better fit the requirements of modern applications. Adaptivity offers a way to design data structures that automatically take advantage of features of the underlying hardware, specific structure and biases in their usage, or side-information, and the limits of data structure adaptivity pose deep research questions. The goal of this seminar was to reflect on these complementary aspects of data structure research and to identify promising research questions. The program provides a snapshot of the current state of research and establishes possible future directions for the field.

Seminar May 4–9, 2025 – http://www.dagstuhl.de/25191
2012 ACM Subject Classification Theory of computation → Data structures design and analysis; Theory of computation → Design and analysis of algorithms; Theory of computation → Parallel algorithms
Keywords and phrases Data structures, Algorithms, Big data, Computational models
Digital Object Identifier 10.4230/DagRep.15.5.1

1 Executive summary

Michael A. Bender (Stony Brook University, US)
John Iacono (ULB – Brussels, BE)
László Kozma (FU Berlin, DE)
Eva Rotenberg (Technical University of Denmark – Lyngby, DK)

License Creative Commons BY 4.0 International license
© Michael A. Bender, John Iacono, László Kozma, and Eva Rotenberg

About the seminar

Data structures are fundamental building blocks of computing systems and are deployed in applications of increasing size and sophistication. The design and analysis of *scalable* data structures has been a central goal of computer science from the beginnings of the field. Yet, the study of data structures remains as active as ever, as it reflects changes in the underlying computational infrastructure, as well as the evolving needs of applications.

[*] Editor / Organizer
[†] Editorial Assistant / Collector

Except where otherwise noted, content of this report is licensed under a Creative Commons BY 4.0 International license

Adaptivity is a multi-faceted goal that can refer to: data structures seamlessly taking advantage of resources offered by the underlying hardware, or restrictions thereof; the usage of application-specific insights or side-information to improve data structure efficiency; the extent to which the data generating process can *adversarially adapt* to the data structure.

Some of these aspects pose long-standing open questions that continue to inspire research (e.g., the *dynamic optimality conjecture* asks whether binary search trees can optimally *adapt* to their usage pattern). More recent directions of adaptivity to side-information, e.g., from machine learning advice, pose fresh research questions, and can also relate to past work on data structures that are locality-adaptive, robust to errors, or that make use of imprecise or unreliable comparisons. A refined notion related to adversarial adaptivity is that of *history-independence*: the question of whether data structures can avoid leaking information about their past sequence of operations. Besides the clear motivation from the point of view of security and privacy, this concept has recently played a key role in designing state-of-the-art data structures for the prototypical list labeling problem. Participants in the seminar have further explored to what extent history independence can be reconciled with optimal efficiency in fundamental data structures such as priority queues.

This seminar was the 16th in a series of loosely related Dagstuhl Seminars on data structures. It brought together a diverse group of researchers with complementary expertise and varying levels of seniority, to illuminate different aspects of data structure scalability and adaptivity. We also aimed to include researchers who are experts in adjacent, applied areas (data structures in computational geometry, data structures for strings, etc.), to inspire and inform each other and to identify relevant research directions.

The organizers of the seminar put a strong emphasis on discussion and collaboration. Participants were asked to give short (10-15 minutes) or medium (20-25 minutes) talks. Most participants spoke, and the format allowed everyone who wished to speak to do so. We explicitly encouraged forward-thinking talks that include open problems, interesting new directions or challenges, focusing on informal, open-ended discussion, rather than polished presentations of past work. In the program we made a deliberate effort to leave space for loosely structured discussion and collaboration, reserving most of the afternoons for this purpose, and having brief status-report sessions where we all met to discuss what people had been discussing.

Topics

The presentations and discussions covered a broad range of topics in classical data structures and diverse applications, as well as new directions inspired by various aspects of scalability and adaptivity.

Models of computation were an important theme of the seminar. Afshani (Section 4.10) discussed possible separations concerning I/O operations and total work, Bille (Section 4.1) discussed data structures in the ultra-wide word RAM model, and Meyer (Section 4.2) talked about unstructured parallel I/O on SSDs in the context of graph algorithms.

Parallelism and distributed computing were also the focus of further talks: King (Section 4.4) presented work on a problem in distributed setting with many Byzantine participants, and Chechik (Section 5.8) discussed the maximum independent set problem in the congested clique model.

Several talks focused on questions (old and new) on graph algorithms: Wein (Section 4.8) talked about a new concept of a "DAG cover" extending classical notions of tree covers, Liu (Section 4.9) discussed a batch version of the dynamic connectivity problem, van der Hoog

(Section 5.2) raised a question about greedy chain decomposition on DAGs, Bilò (Section 4.15) revisited the problem of single-source shortest p-disjoint paths, Gudmundsson (Section 4.16) talked about approximate distance oracles for a special class of sparse graphs, and Fineman (Section 4.17) talked about open questions about incremental topological sorting.

Relating to online problems, Agrawal (Section 5.6) discussed results and open questions concerning a scheduling problem with DAG constraints, and Munro (Section 4.11) talked about a (paired) variant of the paging problem.

On compression and compact data structures, Navarro talked about dynamic bitvectors (Section 4.5) and on a graph problem related to grammar-based compression (Section 5.9), and Gawrychowski (Section 5.4) discussed open problems related to Lempel-Ziv compression.

The analysis of data structures and related notions of adaptivity were a key focus of the seminar. Conway (Section 4.3) gave an overview of results and open questions on history-independent priority queues, while Iacono (Section 5.1) discussed the working set property of priority queues in conjunction with efficient decrease-key. Brodal (Section 4.6) highlighted a simple and efficient data structure for storing integers, and Tarjan (Section 5.3) drew attention to the different analyses of path compression, raising the question of an alternative proof; Kuszmaul (Section 4.14) talked about recent results and open questions on quadratic probing hash tables. Kozma (Section 5.10) pointed out a remaining open question in the analysis of classical selection algorithms, and Farach-Colton (Section 5.5) raised a question related to the Karp-Rabin fingerprinting algorithm.

In his talk, Sedgewick (Section 4.13) advocated for an "algorithms science", using cardinality estimation as a case study. Zamir (Section 4.7) presented recent work on using cryptographic techniques to design asymptotically faster algorithms. Bercea (Section 4.18) discussed a possible new model for Bloom filters with predictions. Storandt (Section 5.7) presented data structure questions arising in the context of geometric intersection problems. Goodrich (Section 4.12) presented computational geometry problems in a setting with probabilistic errors, and raised the question of designing data structures that are robust to probabilistic deletions.

Collaboration was structured around informal working groups, with focus on some of the concrete open problems identified during the seminar. These notably included questions on simultaneous work and I/O optimality (see Section 4.10), fully dynamic graph connectivity (see Section 4.9), history independent heaps (see Section 4.3), heaps with the working set property (see Section 5.1) and greedy chain decompositions of DAGs (see Section 5.2). Several promising directions were explored, and results were obtained already during the seminar. It is expected that the collaborations started at the seminar will lead to publications.

Final Thoughts

The organizers would like to thank the Dagstuhl team for their continuous support and also thank all participants for their contributions to the seminar. The current (16th) seminar was part of a long-running series that has co-evolved with the field, both tracking and inspiring its trends and developments throughout the years.

The seminar fills a unique niche in the computer science research landscape. Although data structural topics are well represented at major venues of theoretical computer science, this series provides essentially the only opportunity for core data structures researchers from around the world to meet and exchange ideas in an informal, collaborative setting, having data structures as the primary focus. (Some other fields, such as computational geometry have a broader tradition of specialized workshops focusing on problem-solving.)

The appreciation by the community for the seminar and the opportunity it offers to personally meet and interact is reflected by the very strong response to invitations and the highly positive subsequent feedback. Respondents particularly praised the relaxed, informal atmosphere and the focus on short talks and loosely structured collaboration around open questions, validating the organizers' efforts in this direction.

Earlier seminars in the series had few female participants. An important focus of the last three seminars was to significantly increase female attendance. In the current seminar, 43% of the invited participants were female, resulting in a 35% female attendance.

Another important goal of the seminar is to encourage the interaction between senior and early career researchers, the latter comprising around a third of the invited participants and eventual attendees. We also made a deliberate effort to include researchers who have not attended the previous seminar(s), these making up a slight majority of both the invitee- and participant lists; we find this to be particularly important for a long-running seminar.

All the aspects mentioned were strongly appreciated by the participants in the post-seminar survey. The survey respondents also praised the coverage of a diverse range of research topics, and the continued efforts by the organizers to refine and improve the seminar format.

2 Table of Contents

3 Seminar program

Sunday May 4, 2025

18:00	*Dinner buffet*

Monday May 5, 2025

7:30	*Breakfast*
9:00	*Opening & Introductions*
9:45	*Coffee break*
10:30	*Short presentations (10-15 minutes max.)*
	Philip Bille, *Ultrawide word RAM*
	Ulrich Meyer, *Utilizing latency on parallel operations on SSDs*
	Alex Conway, *History independent priority queues*
	Valerie King, *Distributed collaboration to share the cost of obtaining external data even when the fraction of Byzantine players is large*
	Gonzalo Navarro, *Dynamic representation of bitvectors*
	John Iacono, *Working set heaps with decrease-key*
	Gerth Stølting Brodal, *Simple data structures with slightly worse bounds*
12:15	*Lunch*
14:00	Medium presentations (20-25 minutes + questions)
	Or Zamir, *Improved runtimes using cryptography*
	Nicole Wein, *Covering approximate shortest paths with DAGs*
	Quanquan Liu, *Monte Carlo batch-dynamic connectivity*
15:30	*Coffee & Cake*
16:00	Collaboration time
18:00	*Dinner*

Tuesday May 6, 2025

7:30	*Breakfast*
9:30	*Short presentations (10-15 minutes max.)*
	Peyman Afshani, *Separations between I/O and work*
	Ian Munro, *Minimizing cache misses with repeated data*
	Ivor van der Hoog, *Greedy chain decomposition on DAGs*
	Robert Endre Tarjan, *On the recurrence in Top-down analysis of path compression by Seidel and Sharir*
	Pawel Gawrychowski, *Two problems on Lempel-Ziv compression*
	Michael Goodrich, *Computational geometry with probabilistically noisy primitive errors*
10:30	*Coffee break*
11:00	Medium presentations (20-25 minutes + questions)
	Bob Sedgewick, *Cardinality estimation, a poster child for algorithm science*
	William Kuszmaul, *Quadratic probing hash tables*
12:15	*Lunch*
13:30	Collaboration time
15:30	*Coffee & Cake and status update*
18:00	*Dinner*

Wednesday May 7, 2025

7:30	*Breakfast*
9:00	*Short presentations*
	Martin Farach-Colton, *Detecting collisions in Karp-Rabin fingerprinting*
	Kunal Agrawal, *Scheduling parallel jobs with a DAG of constraints*
	Sabine Storandt, *Some intersection problems*
10:15	*Coffee break*
10:45	*Short presentations*
	Shiri Chechik, *Maximum independent set in the congested clique model*
	Davide Bilò, *Single-source shortest p-disjoint paths; Fast computation and sparse preservers*
	Joachim Gudmundsson, *Approximate distance oracles for lambda-low density graphs*
12:00	*Group picture*
12:15	*Lunch*
14:00	*Hike* (Lake Noswendl)
15:30	*Coffee & Cake*
18:00	*Dinner*

Thursday May 8, 2024

07:30	*Breakfast*
9:00	*Short presentations*
	Jeremy Fineman, *Incremental topological sort*
	Gonzalo Navarro, *A graph problem with applications to grammar compression*
	Ioana Oriana Bercea, *Oracles for Bloom filters with predictions*
	László Kozma, *Median of medians selection*
10:30	*Coffee break*
11:00	*Collaboration time*
12:15	*Lunch*
13:30	*Collaboration time*
15:30	*Coffee & Cake and status update*
18:00	*Dinner*

Friday May 9, 2023

7:30	*Breakfast and Check-out*
9:00	*Wrap-up session: outcomes of discussions and future directions*
10:30	*Coffee Break*
12:15	*Lunch*

4 Overview of Talks

4.1 Data Structures on the Ultra-Wide Word RAM

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

License Creative Commons BY 4.0 International license
© Philip Bille

We give an overview of the ultra-wide word RAM and recent data structural results in it.

4.2 Rethinking unstructured (parallel) I/O for SSDs

Ulrich Carsten Meyer (Goethe University – Frankfurt am Main, DE)

License Creative Commons BY 4.0 International license
© Ulrich Carsten Meyer

In my talk I highlighted the differences between I/O accesses on traditional hard disks versus flash memory. It appears that there is a potential to use hidden parallelism in many algorithms in order to achieve better performance in the context of unstructured I/O patterns – for example when graph algorithms like SSSP perform random accesses to adjacency lists, which normally only fill a small fraction of the available block size each.

4.3 History independent priority queues

Alexander Conway (Cornell Tech – New York, US)

License Creative Commons BY 4.0 International license
© Alexander Conway

History-independence is a property of some data structures by which their memory layout does not reveal any information about the history of specific operations that lead to its current state. Here we consider this concept applied to priority queues, and ask how efficient a history-independent priority queue can be made. We give a short overview of the best known bounds.

4.4 Distributed collaboration to share the cost of obtaining external data even when the fraction of Byzantine players is large

Valerie King (University of Victoria, CA)

License Creative Commons BY 4.0 International license
© Valerie King
Joint work of Valerie King, John Augustine, Soumyottam Chatterjee, Valerie King, Manish Kumar, Shachar Meir, David Peleg
Main reference John Augustine, Soumyottam Chatterjee, Valerie King, Manish Kumar, Shachar Meir, David Peleg: "Distributed Download from an External Data Source in Faulty Majority Settings", CoRR, Vol. abs/2412.19649, 2024.
URL https://doi.org//10.48550/ARXIV.2412.19649

We consider the Download problem in the Data Retrieval Model, introduced in (DISC'24), where a distributed set of peers, some of which may be Byzantine, seek to learn n bits of data stored at a trustworthy external data source. Each bit of data can be learned by a peer either through a direct (costly) query of the source or through other peers that have already learned it; the goal is to design a collaborative protocol that reduces the maximum number of bits queried by any one peer ("query complexity"). We achieve optimal query complexity in a synchronous fully connected network with resilience to any constant fraction $\beta < 1$ of Byzantine peers, under varying assumptions regarding time and message size.

4.5 Dynamic representation of bitvectors

Gonzalo Navarro (University of Chile – Santiago de Chile, CL)

License Creative Commons BY 4.0 International license
© Gonzalo Navarro
Main reference Gonzalo Navarro: "Adaptive Dynamic Bitvectors", in Proc. of the String Processing and Information Retrieval – 31st International Symposium, SPIRE 2024, Puerto Vallarta, Mexico, September 23-25, 2024, Proceedings, Lecture Notes in Computer Science, Vol. 14899, pp. 204–217, Springer, 2024.
URL https://doi.org//10.1007/978-3-031-72200-4_16

I presented a technique to maintain a bitvector $B[1..n]$ within $n + o(n)$ bits of space, so that operations access, rank, and select, as well as updates to the bitvector (write position, insert bit, delete bit) can all be solved within $O(\log(n/q)/\log\log n)$ amortized time, when there are q queries per update on average. This turns out to be optimal in the cell probe model. This basic tool can be used to obtain similar results on many compact data structures that build on bits. I also showed some experimental results on an implementation of this technique, which show that it is indeed practical. I think the result can be useful for other scenarios, and the main ideas can be inspiring.

4.6 A Simple Integer Successor-Delete Data Structure

Gerth Stølting Brodal (Aarhus University, DK)

License Creative Commons BY 4.0 International license
© Gerth Stølting Brodal
Main reference Gerth Stølting Brodal: "A Simple Integer Successor-Delete Data Structure", in Proc. of the 23rd International Symposium on Experimental Algorithms, SEA 2025, July 22-24, 2025, Venice, Italy, LIPIcs, Vol. 338, pp. 8:1–8:16, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025.
URL https://doi.org//10.4230/LIPICS.SEA.2025.8

A very simple decremental data structure for maintaining a set of integers was presented, that supports initializing the set followed by d deletions and s successor queries in arbitrary order in total time $O(n + d + s(1 + \log_{\max(2,s/n)} \min(s, n)))$. The data structure consists of a single array of integers. The data structure is essentially a special case of the classic union-find data structure with path compression but with unweighted linking (i.e., without linking by rank or size).

4.7 Improving Algorithmic Efficiency using Cryptography

Or Zamir (Tel Aviv University, IL)

License 🅭 Creative Commons BY 4.0 International license
© Or Zamir
Joint work of Vinod Vaikuntanathan, Or Zamir
Main reference Vinod Vaikuntanathan, Or Zamir: "Improving Algorithmic Efficiency using Cryptography", CoRR, Vol. abs/2502.13065, 2025.
URL https://doi.org//10.48550/ARXIV.2502.13065

Cryptographic primitives have been used for various non-cryptographic objectives, such as eliminating or reducing randomness and interaction. We show how to use cryptography to improve the time complexity of solving computational problems. Specifically, we show that under standard cryptographic assumptions, we can design algorithms that are asymptotically faster than existing ones while maintaining correctness in a black-box manner.

4.8 Covering Approximate Shortest Paths with DAGs

Nicole Wein (University of Michigan – Ann Arbor, US)

License 🅭 Creative Commons BY 4.0 International license
© Nicole Wein
Joint work of Sepehr Assadi, Gary Hoppenworth, Nicole Wein
Main reference Sepehr Assadi, Gary Hoppenworth, Nicole Wein: "Covering Approximate Shortest Paths with DAGs", in Proc. of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025, pp. 2269–2280, ACM, 2025.
URL https://doi.org//10.1145/3717823.3718298

I will talk about our newly defined notion of a "DAG cover". It is a directed analog of a tree cover, which is closely related to a probabilistic tree embedding. A DAG cover of a general directed graph G is a small collection of DAGs so that for all pairs of vertices s,t, some DAG in the collection provides low distortion for $\text{dist}(s,t)$. I will discuss upper and lower bounds for DAG covers in various parameter regimes, and pose some open problems.

4.9 Monte Carlo Batch-Dynamic Connectivity

Quanquan C. Liu (Yale University – New Haven, US) and Valerie King (University of Victoria, CA)

License 🅭 Creative Commons BY 4.0 International license
© Quanquan C. Liu and Valerie King
Joint work of Lalia Cann, Valerie King, Quanquan C. Liu

Connectivity is an important problem with countless real-world applications. In particular, the graphs we perform connectivity queries on are often *dynamic* where edges are inserted or deleted with high frequency. The dynamic connectivity problem then seeks to answer connectivity queries while an online sequence of edge insertions and deletions occur in the graph. In this paper, we give the first parallel batch-dynamic implementation with provably large efficiency and parallelism guarantees. Furthermore, our guarantees show that the runtimes are small, even in the worst case. To do this, we simplify, adapt, and parallelize the sequential Monte Carlo algorithms of Kapron, King, and Mountjoy [SODA 2013], Gibb, Kapron, King, and Mountjoy [2015], and Wang for the batch-dynamic setting. These algorithms provably require very small, $\text{poly}(\log n)$ time per update and query *even in the* *worst-case.* Our simplification uses only Euler Tour (ET) trees, rather than more complicated path compression data structures, making it simultaneously simpler to implement and faster than more complicated structures.

We present an open problem regarding whether we can eliminate the level data structure altogether in both the sequential dynamic algorithm and our algorithm.

4.10 Simultaneous Work And I/O Optimality

Peyman Afshani (Aarhus University, DK)

License Creative Commons BY 4.0 International license
© Peyman Afshani

We study a number of algorithmic problems in the I/O model of computation, aiming to obtain an algorithm which achieves the optimal number of I/O operations in the I/O model as well as achieving the optimal amount of CPU work in the classical RAM model. In some cases, this is already known to be impossible (Afshani, Brodal, Sitchinava, manuscript, 2025). However, if that is the case, the goal would be to obtain an optimal work-I/O trade-off. The problems that we study are the following: deferred data structures and red-blue sorting.

For the first problem, the input is an unsorted list of values which needs to be stored for online queries. However, the total number of queries is not known and the goal is to minimize the total amount of preprocessing time done. It is a classical result of Karp and Motwani (SIAM JoC'88) that if the total number of queries is k (which is unknown in advance), then this can be done in $O(n \log k)$ time for a number of different queries such as predecessor search queries. A similar performance can be achieved in the I/O model however, achieving both seems to be impossible and thus this leads to work-I/O trade-off for this problem.

For the second problem, the input is a set of n values, each associated with one of the two colors, red or blue. The goal is to sort the items using the minimum number of comparisons, however, for two elements u and w if there is no other element of the opposite color that lies between them, then the determining the order of u and w is not necessary. When there is only one value v of one color, then the problem reduces to simply the splitting the values with respect to pivot v which can be done in linear time, as well as linear I/Os and thus simultaneous work and I/O optimality is possible in this case. It has been observed that in the worst-case, this can be done in general. However, obtaining an adaptive bound on the number of comparisons is a more interesting problem for which obtaining such a simultaneous optimality does not seem possible.

4.11 Minimizing Cache Misses with Repeated Data: The Cache Pair Problem

Ian Munro (University of Waterloo, CA)

License Creative Commons BY 4.0 International license
© Ian Munro
Joint work of Ian Munro, Amin Khodaee

We consider the generalization of the problem of minimizing cache misses in the situation in which data items are stored on multiple cache lines across memory (i.e. there are multiple copies of the same objects). The issue is, then what line to evict when a new one must be uploaded. In the usual (single copy) situation, it is well known that (i) the minimum number of cache misses for the off line problem (i.e. all requests are known in advance) can easily be determined and (ii) in the on line case, one can come within a factor of two of this bound given twice the number of cache lines. We show that if each datum is stored on two cache lines (so we have a “choice” of which line to bring in,) the off line version of the problem is NP-hard, and indeed, assuming the unique games conjecture, coming within twice the optimal using twice the given number of line is also NP-hard. We give an easy solution to the on line version that is within four times as many cache misses as the optimal given quadruple the space.

4.12 Computational geometry with probabilistically noisy primitive errors

Michael T. Goodrich (University of California – Irvine, US)

License Creative Commons BY 4.0 International license
© Michael T. Goodrich
Joint work of David Eppstein, Michael T. Goodrich, Vinesh Sridhar
Main reference David Eppstein, Michael T. Goodrich, Vinesh Sridhar: “Computational Geometry with Probabilistically Noisy Primitive Operations”, in Proc. of the 19th International Symposium on Algorithms and Data Structures, WADS 2025, August 11-15, 2025, York University, Toronto, Canada, LIPIcs, Vol. 349, pp. 24:1–24:20, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025.
URL https://doi.org//10.4230/LIPICS.WADS.2025.24

Much prior work has been done on designing computational geometry algorithms that handle input degeneracies, data imprecision, and arithmetic round-off errors. We take a new approach, inspired by the noisy sorting literature, and study computational geometry algorithms subject to noisy Boolean primitive operations in which, e.g., the comparison “is point q above line L?” returns the wrong answer with some fixed probability. We propose a novel technique called path-guided pushdown random walks that generalizes the results of noisy sorting. We apply this technique to solve point-location, plane-sweep, convex hulls in 2D and 3D, dynamic 2D convex hulls, and Delaunay triangulations for noisy primitives in optimal time with high probability.

4.13 Cardinality Estimation: A Poster Child for Algorithm Science

Robert Sedgewick (Princeton University, US)

License Creative Commons BY 4.0 International license
© Robert Sedgewick
Joint work of Svante Janson, Jérémie O. Lumbroso, Robert Sedgewick
Main reference Svante Janson, Jérémie O. Lumbroso, Robert Sedgewick: “Bit-Array-Based Alternatives to HyperLogLog”, in Proc. of the 35th International Conference on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms, AofA 2024, June 17-21, 2024, University of Bath, UK, LIPIcs, Vol. 302, pp. 5:1–5:19, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.
URL https://doi.org//10.4230/LIPICS.AOFA.2024.5

This talk emphasizes the role of algorithm science in the decades-long development of state-of-the-art cardinality estimation algorithms, from the seminal papers of Flajolet and coauthors to the recent bit-array based methods of Lumbroso, Janson and Sedgewick.

4.14 Quadratic probing hash tables

William Kuszmaul (Carnegie Mellon University – Pittsburgh, US)

License Creative Commons BY 4.0 International license
© William Kuszmaul
Joint work of William Kuszmaul, Zoe Xi
Main reference William Kuszmaul, Zoe Xi: "Towards an Analysis of Quadratic Probing", in Proc. of the 51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia, LIPIcs, Vol. 297, pp. 103:1–103:19, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.
URL https://doi.org//10.4230/LIPICS.ICALP.2024.103

Since 1968, one of the simplest open questions in the theory of hash tables has been to prove anything nontrivial about the correctness of quadratic probing. We make the first tangible progress towards this goal, showing that there exists a positive-constant load factor at which quadratic probing is a constant-expected-time hash table. Our analysis applies more generally to any fixed-offset open-addressing hash table, and extends to higher load factors in the case where the hash table examines blocks of some size $B = \omega(1)$.

4.15 Single-source shortest p-disjoint paths: fast computation and sparse preservers

Davide Bilò (University of L'Aquila, IT)

License Creative Commons BY 4.0 International license
© Davide Bilò
Joint work of Davide Bilò, Gianlorenzo D'Angelo, Luciano Gualà, Stefano Leucci, Guido Proiett, Mirko Rossi
Main reference Davide Bilò, Gianlorenzo D'Angelo, Luciano Gualà, Stefano Leucci, Guido Proietti, Mirko Rossi: "Single-Source Shortest p-Disjoint Paths: Fast Computation and Sparse Preservers", in Proc. of the 39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference), LIPIcs, Vol. 219, pp. 12:1–12:21, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
URL https://doi.org//10.4230/LIPICS.STACS.2022.12

Let G be a directed graph with n vertices, m edges, non-negative edge costs, and a fixed source vertex s. Given a target vertex t, the problem of computing a set S_t of p pairwise edge-disjoint (simple) paths from s to t in G having minimum total cost (if such paths exist) can be solved in $O(p(m + n \log n))$ time using the Successive Shortest Path algorithm. We are interested in constructing a compact data structure that, when queried with any target vertex t, can quickly return S_t.

For $p = 1$, we can precompute a shortest-path tree of G rooted at s and build a data structure of size $O(n)$ that answers queries in time linear in the size of the output. The preprocessing time is $O(m + nlogn)$ when using Dijkstra's algorithm. For $p = 2$, Suurballe and Tarjan [Networks 1984] designed a data structure with the same asymptotic trade-offs among preprocessing time, query time, and size. For general values of p, Bilò et al. [STACS 2022] designed a $O(pn(pn + n \log n))$-time algorithm that computes a subgraph H of G of size at most $p(n - 1)$ containing S_t for every target vertex t. The size of H is optimal when G is p-edge-outconnected from s. This result implies the existence of a data structure with preprocessing time $O(pn(pn + n \log n))$, size $O(pn)$, and query time $O(p(pn + n \log n))$. Improving the preprocessing and query times remains an interesting open problem.

4.16 A well-separated pair decomposition for low density graphs

Joachim Gudmundsson (The University of Sydney, AU)

License 🅭 Creative Commons BY 4.0 International license
© Joachim Gudmundsson
Joint work of Joachim Gudmundsson, Sampson Wong

Low density graphs are considered to be a realistic graph class for modelling road networks. It has advantages over other popular graph classes for road networks, such as planar graphs, bounded highway dimension graphs, and spanners. We believe that low density graphs have the potential to be a useful graph class for road networks, but until now, its usefulness is limited by a lack of available tools. In this talk, we show two new fundamental tools for low density graphs, that is, a well-separated pair decomposition and an approximate distance oracle. We believe that by expanding the algorithmic toolbox for low density graphs, we can help provide a useful and realistic graph class for road networks, which in turn, may help explain the many efficient and practical heuristics available for road networks.

4.17 Incremental topological sort

Jeremy Fineman (Georgetown University – Washington, DC, US)

License 🅭 Creative Commons BY 4.0 International license
© Jeremy Fineman

This talk gives a short overview on incremental topological sort. Here m edges are added to an n-vertex graph incrementally, and the goal is to maintain a topological sort. The current state of the art includes the following total update times, ignoring log factors:

- Dense graphs (large m): $\tilde{O}(n^2)$ from Bender, Fineman, Gilbert, and Tarjan. This work also includes an extension to maintain a strongly connected components (SCCs).
- Sparse graphs (small m): $\tilde{O}(m^{4/3})$ from Battacharya and Kulkarni. Bernstein, Dudeja, and Pettie later obtained the same bound for SCCs.

Finally, Chen, Kyng, Liu, Meierhans, and Probst Gutenberg obtained an algorithm that has total update time of $m^{1+o(1)}$ for the related problem of cycle detection, but their algorithm does not maintain a topological sort. Moreover, their algorithm is not combinatorial.

We have reason to believe that it should be possible to obtain a bound of $\tilde{O}(n\sqrt{m})$ for incremental topological sort with a combinatorial algorithm. Ignoring log factors, this would be at least as good as the best dense algorithm for all m and n, and it would beat the best sparse algorithm for graphs of sufficient density.

4.18 Data structures and predictions

Ioana-Oriana Bercea (KTH Royal Institute of Technology – Stockholm, SE)

License 🅭 Creative Commons BY 4.0 International license
© Ioana-Oriana Bercea

The Daisy Bloom filter is an approximate memembership data structure (filter) that has access to knowledge of the distribution of the input and query distribution. While we understand how to set optimal parameters in this setting, this has yet to translate into practical gains. We highlight some questions in this direction, with the exciting prospect of bridging the gap between classical results on instance optimality and emerging directions such as algorithms with predictions.

5 Open problems

5.1 Working set heaps with decrease-key

John Iacono (ULB - Brussels, BE)

License Creative Commons BY 4.0 International license
© John Iacono

Priority queues are one of the oldest abstract data types in computer science, and those that support the decrease-key operation, such a Fibonacci Heaps, have been instrumental in a number of efficient graph algorithms such as single source shortest paths (SSSP). Recent progress in the beyond-worst-case analysis of SSSP has shown the utility of having a priority queue that supports constant time-decrease key as well as the working set property, one variant of which requires that the cost of an extract-min be proportional to the logarithm of the number of operations performed on the heap while that data item was present in it. No such pointer-model data structure exists with both the working-set property and constant-time decrease-key. Also briefly discussed were some variants of what the working set could mean for priority queues.

5.2 Greedy chain decomposition on DAGs

Ivor van der Hoog (Technical University of Denmark – Lyngby, DK)

License Creative Commons BY 4.0 International license
© Ivor van der Hoog

Given a directed graph G with n vertices and m edges, we consider the problem of computing a *longest directed path decomposition*: a recursive partitioning of the graph into sets of vertices, where each set consists of the vertices along some longest directed path in the remaining graph. Observe that, since the longest path need not be unique, the decomposition is not unique either.

We present a simple two-stage algorithm that computes such a decomposition in time $O(nm)$, improving over the previous best-known bound of $O(n^{2.5})$ when the number of edges m is subquadratic. The algorithm proceeds as follows:

- First, perform a DFS to find a longest directed path in $O(m)$ time. If its length is at least n, remove its vertices and recurse.
- Otherwise, the graph has height at most n. We compute a height decomposition and construct a flow-DAG by connecting a super-source to all sources in G, and a super-sink to all vertices of maximum height. A blocking flow in this DAG can be computed in $\tilde{O}(m)$ time, and its removal eliminates all longest paths. Since the height decreases by at least one per iteration and is initially at most $\sqrt{n}$, this phase takes $\tilde{O}(m\sqrt{n})$ time overall.

5.3 Top-down analysis of path compression

Robert Endre Tarjan (Princeton University, US)

License 🅭 Creative Commons BY 4.0 International license
© Robert Endre Tarjan
Main reference Raimund Seidel, Micha Sharir: "Top-Down Analysis of Path Compression", SIAM J. Comput., Vol. 34(3), pp. 515–525, 2005.
URL https://doi.org//10.1137/S0097539703439088

Seidel and Sharir gave a proof of the inverse-Ackermann-function upper bound for path compression based on a beautiful top-down recurrence. The analysis does not use Ackermann's function explicitly, but uses a related function they call "J." Problem: Give a simple, direct proof of the inverse-Ackermann function bound using their top-down recurrence and the classical definition of Ackermann's function.

5.4 Two problems on Lempel-Ziv compression

Paweł Gawrychowski (University of Wroclaw, PL)

License 🅭 Creative Commons BY 4.0 International license
© Paweł Gawrychowski

We ask two questions about the widely used Lempel-Ziv compression algorithm, and give an overview of some known results relating to these questions:

- Can pattern matching on a Lempel-Ziv compressed text be done in $O(n+m)$ time, for a string of compressed length n and a pattern of length m?
- Can we design a data-structure using $O(n)$ space which allows for $O(\log N)$-time indexing of characters in a string of length N with a Lempel-Ziv compressed representation of length n?

5.5 Detecting collisions in Karp-Rabin fingerprinting

Martin Farach-Colton (NYU – New York, US)

License 🅭 Creative Commons BY 4.0 International license
© Martin Farach-Colton

The Karp-Rabin Fingerprint method is a clean and simple way to achieve fast algorithms for many string matching problems. The idea is to treat the characters of a string as the coefficients of a polynomial and to take the value of this polynomial evaluated at Σ, the size of the alphabet. In order to keep the number of bits down (and thus to be able to perform constant-time manipulations), the value is computed modulo a prime number whose value is some polynomial of n, the size of the longest string under consideration. For any substring, this modular evaluation of the induced polynomial is the Karp-Rabin fingerprint of that substring.

Although taking the mod does speed up the fingerprinting algorithm, it induces collisions where two substrings of the same length might evaluate to the same fingerprint even though they do not match.

The open question is: how quickly can one take a string and a prime and check of any pair of substrings (of the same length) of the string have colliding fingerprints? In other words, are there two distinct substrings for which the given prime induces a collision? A quadratic-time algorithm is straightforward using suffix trees. Can we achieve a linear time algorithm or at least an $O(n \log n)$-time algorithm?

It's ok if we reject some polynomials that don't, in fact, induce a collision, as long as we do it with small probability.

5.6 Scheduling parallel jobs with a DAG of constraints

Kunal Agrawal (Washington University – St. Louis, US)

License Creative Commons BY 4.0 International license
© Kunal Agrawal

We give an overview of known results and open questions regarding the scheduling of parallel tasks, each of which is represented as a directed acyclic graph of precedence constraints (thus allowing for both parallelism between tasks as well as intra-task parallelism).

5.7 Some Intersection Problems

Sabine Storandt (Universität Konstanz, DE)

License Creative Commons BY 4.0 International license
© Sabine Storandt

The talk introduces the following problems in the realm of intersection reporting data structures:

- Construct an efficient segment-circle intersection data structure under the assumption that the n input segments form a connected graph. The general data structure requires $O(n^{1.5})$ preprocessing time and $O(n^{0.5} + k)$ query time where k is the output size.
- Given a set of geometric objects, preprocess it such that for a given query range pairs of input objects intersection inside the query range can be reported efficiently. While for many types of objects and ranges, 3-SUM hardness can be shown, half-plane queries for segments and lines escape that lower bound. However, faster query times might be possible in that case and more general scenarios as the input objects being polylines is wide open.

5.8 Maximum independent set in the congested clique model

Shiri Chechik (Tel Aviv University, IL)

License Creative Commons BY 4.0 International license
© Shiri Chechik

We ask how many rounds of communication are necessary to compute a maximum independent set of a graph G in a particular model of distributed computation, known as the congested clique model. In this model there are n players identified with the vertices of G, each player initially knows the vertices adjacent to itself, and can send b bits of information to every other player in each round of communication.

We give a short overview of the known results.

5.9 A graph problem with applications to grammar compression

Gonzalo Navarro (University of Chile – Santiago de Chile, CL)

License Creative Commons BY 4.0 International license
© Gonzalo Navarro

One can represent a text $T[1..u]$ with a context-free grammar of size n that produces (only) T. This representation requires $O(n \log n)$ bits. However, every structure providing random access to T using the grammar requires $\Theta(n \log u)$ further bits of space, to record the expansion length of the nonterminals. The question is whether we can remove this term and still access the text efficiently. One choice is to sample $o(n)$ nodes so that the others' expansion lengths can be obtained in polylog time from the sampled ones. This boils down to a problem sparse on acyclic graphs.

5.10 Median of medians in small groups

László Kozma (FU Berlin, DE)

License Creative Commons BY 4.0 International license
© László Kozma

I briefly mentioned an open problem raised by Chen and Dumitrescu in connection to the "median of medians" selection algorithm of Blum, Floyd, Pratt, Rivest, and Tarjan from 1973. This linear time selection algorithm is based on arranging the input elements in groups of five, leading to the natural question of whether groups of three are also sufficient. The classical analysis only yields a superlinear bound in this case. Chen and Dumitrescu point out that contrary to common belief, this does not necessarily imply that the running time with groups of three is indeed superlinear, and constructing a "difficult instance" appears to be difficult; thus, a gap remains in our understanding of this fundamental algorithm.

Participants

- Peyman Afshani
Aarhus University, DK
- Kunal Agrawal
Washington University – St. Louis, US
- Hideo Bannai
Institute of Science Tokyo, JP
- Michael A. Bender
Stony Brook University, US
- Ioana Oriana Bercea
KTH Royal Institute of Technology – Stockholm, SE
- Philip Bille
Technical University of Denmark – Lyngby, DK
- Davide Bilò
University of L'Aquila, IT
- Gerth Stølting Brodal
Aarhus University, DK
- Shiri Chechik
Tel Aviv University, IL
- Alexander Conway
Cornell Tech – New York, US
- Justin Dallant
UL – Brussels, BE
- Aditi Dudeja
Paris Lodron Universität Salzburg, AT
- Faith Ellen
University of Toronto, CA
- Martin Farach-Colton
NYU – New York, US
- Jeremy Fineman
Georgetown University – Washington, DC, US
- Pawel Gawrychowski
University of Wroclaw, PL
- Michael Goodrich
University of California – Irvine, US
- Joachim Gudmundsson
The University of Sydney, AU
- Inge Li Gørtz
Technical University of Denmark – Lyngby, DK
- John Iacono
ULB – Brussels, BE
- Rob Johnson
Broadcom – San Jose, US
- Valerie King
University of Victoria, CA
- Tomasz Kociumaka
MPI für Informatik – Saarbrücken, DE
- Hanna Komlós
NYU – New York, US
- László Kozma
FU Berlin, DE
- William Kuszmaul
Carnegie Mellon University – Pittsburgh, US
- Jingxun Liang
Carnegie Mellon University – Pittsburgh, US
- Quanquan C. Liu
Yale University – New Haven, US
- Ulrich Carsten Meyer
Goethe University – Frankfurt am Main, DE
- Ian Munro
University of Waterloo, CA
- Gonzalo Navarro
University of Chile – Santiago de Chile, CL
- Eva Rotenberg
Technical University of Denmark – Lyngby, DK
- Robert Sedgewick
Princeton University, US
- Marek Sokolowski
MPI für Informatik – Saarbrücken, DE
- Teresa Steiner
Technical University of Denmark – Lyngby, DK
- Sabine Storandt
Universität Konstanz, DE
- Robert Endre Tarjan
Princeton University, US
- Ivor van der Hoog
Technical University of Denmark – Lyngby, DK
- Stefan Walzer
KIT – Karlsruher Institut für Technologie, DE
- Nicole Wein
University of Michigan – Ann Arbor, US
- Huacheng Yu
Princeton University, US
- Or Zamir
Tel Aviv University, IL
- Renfei Zhou
Carnegie Mellon University – Pittsburgh, US