

Discrete Algorithms on Modern and Emerging Compute Infrastructure

Kathrin Hanauer^{*1}, Uwe Naumann^{*2}, Alex Pothén^{*3}, and Robert Schreiber^{*4}

1 Universität Wien, AT. kathrin.hanauer@univie.ac.at

2 RWTH Aachen, DE. naumann@stce.rwth-aachen.de

3 Purdue University – West Lafayette, US. apothén@purdue.edu

4 Cerebras Systems – Palo Alto, US. rob.schreiber@cerebras.net

Abstract

Inspired by three plenary talks by leading figures in the area of “Discrete algorithms on modern and emerging compute infrastructure” this Dagstuhl Seminar emphasized focus sessions and working groups to dive into this very versatile topic. Lively discussions between experts from academia, research laboratories, and industry yielded a number of promising prospects for follow-up activities. As always, Dagstuhl provided the perfect setting for this kind of scientific exchange.

Seminar May 12–17, 2024 – <https://www.dagstuhl.de/24201>

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics; Mathematics of computing → Mathematical software; Computing methodologies → Parallel computing methodologies

Keywords and phrases Combinatorial Scientific Computing, Discrete Algorithms, Graph Algorithms, High-Performance Computing

Digital Object Identifier 10.4230/DagRep.14.5.12

1 Executive Summary

Kathrin Hanauer (Universität Wien, AT)

Uwe Naumann (RWTH Aachen, DE)

Alex Pothén (Purdue University – West Lafayette, US)

Robert Schreiber (Cerebras Systems – Palo Alto, US)

License  Creative Commons BY 4.0 International license

© Kathrin Hanauer, Uwe Naumann, Alex Pothén, and Robert Schreiber

We are happy to report on a lively and productive scientific discourse on discrete algorithms on modern and emerging compute infrastructure. As always, Dagstuhl presented an ideal setting for this kind of interdisciplinary meeting of experts from diverse backgrounds.

The aim was to identify requirements for

1. discrete algorithms imposed by emerging compute infrastructure;
2. emerging compute infrastructure imposed by discrete algorithms;
3. curricula at universities aiming to educate the next generation of designers of novel discrete algorithms as well as of future compute infrastructure.

We focused on sparse linear algebra and graph algorithms while reaching out to a diverse set of representatives from industry combining expertise in modern accelerators, next-generation silicon, and quantum computing.

Research questions addressed included the following:

* Editor / Organizer



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

Discrete Algorithms on Modern and Emerging Compute Infrastructure, *Dagstuhl Reports*, Vol. 14, Issue 5, pp. 12–24

Editors: Kathrin Hanauer, Uwe Naumann, Alex Pothén, and Robert Schreiber



DAGSTUHL
REPORTS Dagstuhl Reports

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

1. How should today's discrete algorithms be re-designed in order to meet the requirements of emerging compute infrastructure?
 - a. Can lessons learned while mapping discrete algorithms onto modern compute infrastructure be (partially) generalized for emerging compute infrastructure?
 - b. What are implications for (combinations of) deterministic, stochastic, and data-driven methods?
 - c. What impact on the design of discrete algorithms and their implementation can be expected from likely hierarchy / heterogeneity in emerging compute infrastructure?
2. How can emerging compute infrastructure be tailored towards the needs of practically relevant discrete problems and their algorithmic solution?
 - a. How do we support irregularity and dynamics inherent in sparse linear algebra and graph problems by suitable hardware architecture / system software?
 - b. What do suitable programming models / languages look like?
 - c. How could we account for memory-boundedness?

Following individual 5 min introductions, the program consisted of three plenary talks, four plenary focus sessions / panel discussions, and four non-plenary working groups. Refer to the respective abstracts for details.

2 Table of Contents

Executive Summary

Kathrin Hanauer, Uwe Naumann, Alex Pothén, and Robert Schreiber 12

Overview of Talks

Challenges for Computational Graph Algorithms
John Gilbert 15

Graph Algorithms in Unsettled Times
Alex Pothén 15

Trail Guide To Parameterized Algorithms In Practice
Blair D. Sullivan 15

Working groups

Single-instance vs. batched vs. sequence of problems
Paolo Bientinesi 16

Dynamic Algorithms Working Group
Kathrin Hanauer, David A. Bader, Oded Green, and Helen Xu 17

Towards a theory of tile-centric computation
Johannes Langguth 18

Adjoint Differentiation and Its Challenges
Johannes Lotz, Martin Bucker, and Paul D. Hovland 19

Hypergraph Algorithms Working Group
Nate Veldt, Alex Crane, Gero Kauer auf, Daniel Král', Henning Meyerhenke, Henrik Reinstädler, Christian Schulz, Blair D. Sullivan, and Bora Uçar 20

Panel discussions

Algorithms: Beyond the Static
Kathrin Hanauer, Quanquan C. Liu, Manuel Penschuck, and Helen Xu 21

The Future of Computing
Bruce Hendrickson, Jakob Engblom, Chris Goodyer, and Oded Green 21

MLQAOA: Graph Learning Accelerated Hybrid Quantum-Classical Multilevel QAOA
Ilya Safro and Eleanor Rieffel 22

Wafer Scale Computing: Fine Grain Parallelism and Rethinking Parallel Computing
Robert Schreiber 23

Participants 24

3 Overview of Talks

3.1 Challenges for Computational Graph Algorithms

John Gilbert (University of California – Santa Barbara, US)

License © Creative Commons BY 4.0 International license
© John Gilbert

Though applications of graphs go back at least to Euler in 1736, the age of large-scale computation with graphs arguably began in the 1970s. Computing efficiently with graphs has always been hard, but the challenges have evolved quite a bit over the past 50 years. This talk speculates on what key challenges the designers and users of high-performance graph computation will face during the next 10 years, organized roughly into the categories: applications; data; machine architecture; algorithms; and productivity.

3.2 Graph Algorithms in Unsettled Times

Alex Pothén (Purdue University – West Lafayette, US)

License © Creative Commons BY 4.0 International license
© Alex Pothén

We live in the time of a changing and uncertain computing environment. The end of Moore's law and Dennard scaling has led to searches for new computing substrates, from chiplets, accelerators, wafer scale chips, neuromorphic computers, quantum computers, etc. The growth of data science has led to graph models for unstructured data of increasing sizes for downstream inference tasks. The artificial intelligence revolution has led to the study of large scale graph neural networks, but also learning augmented algorithms, where machine learning concepts are used to provably improve the quality of the solution or the run time of the algorithm. All of these factors lead to the development of new models for algorithm design, including approximation algorithms, distributed algorithms, online algorithms, semi-streaming algorithms, dynamic algorithms, fixed parameter algorithms, etc. I will survey of some of these topics in this introductory talk. Several of the workshops and panels at this Dagstuhl Seminar will consider these topics in more detail, and my hope is that these discussions could serve as a helpful vade mecum for algorithms researchers.

3.3 Trail Guide To Parameterized Algorithms In Practice

Blair D. Sullivan (University of Utah – Salt Lake City, US)

License © Creative Commons BY 4.0 International license
© Blair D. Sullivan

This talk introduces the audience to a mixture of classic and recent algorithmic techniques which originate primarily in the theoretical computer science community and exploit the non-uniformity of computational hardness. In particular, the focus is on ideas that I think hold promise for real-world network analysis in the next decade – despite often being completely impractical in their current form! I also briefly discuss lessons learned from applications where some of these techniques have been engineered successfully to impact domain science, and highlight what I see as key challenges in the space.

4 Working groups

4.1 Single-instance vs. batched vs. sequence of problems

Paolo Bientinesi (University of Umeå, SE)

License  Creative Commons BY 4.0 International license
 Paolo Bientinesi

Traditionally, library kernels are designed to support one specific mathematical operation. The kernel interface is meant to make it possible to pass input arguments and to set algorithmic parameters. The benefits of such a library design are undeniable and numerous, e.g., separation of concerns, possibility of optimization, readability, and more.

In many scientific applications, not one, but many problems of the same kind have to be solved. When such problems are independent of one another, they can be solved concurrently. This observation led to the development of “batched” operations and respective libraries (sometimes referred to as “streaming”). These are especially beneficial when each individual problem is so small that the overhead due to a function call is noticeable. Batched operations are also of obvious importance for data parallelism.

A more general (and arguably more common) scenario arises when an application involves the solution of multiple problems on the same kind, and the problems are in some way correlated with one another. Examples include multiple linear systems in which the coefficient matrix varies parametrically, or problems that share partially the input data. In this case, we talk about “sequences” of problems. Depending on the nature of the correlation, a sequence of problems can be solved considerably faster than solving each problem individually.

Questions to be discussed:

- Applications and workflows in which batches and sequences of problems arise.
- How are problems correlated? How to exploit the correlation?
- Limitations of the current interfaces.

Target Audience:

Anyone who designs and/or implements computational libraries for mathematical operations.

Report:

The working group consisted of 8-9 people and originated a lively discussion. We first had a round of introductions during which everybody presented their “computational scenario(s)”. We then brainstormed on what it would take for a library interface to capture such scenarios. We quickly identified that in some cases it makes sense to abandon the concept of library calls. One such case occurs when one has to solve not one single large problem, but many problems that exhibit some form of commonality (e.g., similar input data). Another case is when the ordering of the problems is a critical factor, for instance because of the size of the intermediate results. In these cases, we discussed how a compiler (in contrast to a library) would be preferable.

4.2 Dynamic Algorithms Working Group

Kathrin Hanauer (Universität Wien, AT), David A. Bader (NJIT – Newark, US), Oded Green (NVIDIA – Alpharetta, US), and Helen Xu (Georgia Institute of Technology – Atlanta, US)

License © Creative Commons BY 4.0 International license
© Kathrin Hanauer, David A. Bader, Oded Green, and Helen Xu

Traditional, “static” algorithms follow a quite simplistic scheme: Given some input data, they perform a number of computational steps and then stop and produce an output. However, real-world data often is nowhere near static. Instead, it undergoes a constant stream of modifications, caused, e.g., by user interactions, environmental changes, traffic flows, social network activity, stock exchange dealing, and much more. A static algorithm would have to be re-run each time the current result of its computations is needed. Worse, an update to the graph within the above applications also requires that the data structure representing the graph also be update. While the above seems both obvious and trivial, it overlooks the fact that updating these sparse representations is in practice very challenging, especially when wanting to stay as close as possible to CSR (compressed sparse row), which is the de-facto data representation used for sparse applications.

For *dynamic* algorithms there is a need to update both the graph as well as update algorithmic values associated to the problem. In contrast to static algorithms, dynamic algorithms have received much fewer attention, especially in practice. This is in part due to the following facts: 1) static graph problems are challenging in their own right and 2) the need a for high-performing dynamic graph data structure prevents people from tackling these harder problems as to have an effective dynamic graph algorithm one must first ensure that the dynamic graph data structure will not become the bottleneck of the new algorithm. Such bottlenecks can include the operations of updating the graph (aka insert and delete operations) or in the graph access functionality that is necessary for simply accessing the vertices and edges (as might be needed in a graph traversal problem). In this working group, we discussed in particular the following topics:

Whereas de-facto standards exist for the efficient representation of static graphs, such as the compressed sparse row format (CSR), a universal model for dynamic graphs is still in the open. There has been a series of developments, such as STINGER [3], Aspen [2], CPMA [6], cuSTINGER [4], or Hornet [1]. Still, different approaches exist and seem necessary to accommodate for individual use case scenarios. The ideal scenario is finding general-purpose “dynamic CSR“ format, but this is not without challenges¹.

Introductory textbooks and survey papers exist on my algorithmic topics, but relatively little material is available that focuses on dynamic algorithms and particularly their efficient implementation and evaluation. Notably, there is a survey on fully-dynamic graph algorithms [5] with an emphasis on experimental results, which gives an overview over recent results in this area. However, following up on the first discussion, there is a lack of introductory materials describing, e.g., good enough and sufficiently simple data structures for dynamic graphs that can be taught to undergraduate students. Furthermore, a standardized benchmark data set could help to foster empirical research on dynamic algorithms.

The working group had 8–13 participants.

¹ Oded Green described the Hornet data structure has essentially a dynamic version of CSR that allows for sparse matrices and graphs to grow with few memory allocations necessary. The big reason that Hornet can be considered CSR compatible is that memory access patterns are quite similar.

References

- 1 Federico Busato, Oded Green, Nicola Bombieri, and David A Bader. Hornet: An efficient data structure for dynamic sparse graphs and matrices on gpus. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pages 1–7. IEEE, 2018.
- 2 Laxman Dhulipala, Guy E. Blelloch, and Julian Shun. Low-latency graph streaming using compressed purely-functional trees. In Kathryn S. McKinley and Kathleen Fisher, editors, *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Phoenix, AZ, USA, June 22-26, 2019*, pages 918–934. ACM, 2019.
- 3 David Ediger, Robert McColl, E. Jason Riedy, and David A. Bader. STINGER: high performance data structure for streaming graphs. In *IEEE Conference on High Performance Extreme Computing, HPEC 2012, Waltham, MA, USA, September 10-12, 2012*, pages 1–5, 2012.
- 4 Oded Green and David A Bader. cuSTINGER: Supporting dynamic graph algorithms for gpus. In *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–6. IEEE, 2016.
- 5 Kathrin Hanauer, Monika Henzinger, and Christian Schulz. Recent advances in fully dynamic graph algorithms – A quick reference guide. *ACM J. Exp. Algorithmics*, 27:1.11:1–1.11:45, 2022.
- 6 Brian Wheatman, Randal C. Burns, Aydin Buluç, and Helen Xu. CPMA: An efficient batch-parallel compressed set without pointers. In Michel Steuwer, I-Ting Angelina Lee, and Milind Chabbi, editors, *Proceedings of the 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming, PPOPP 2024, Edinburgh, United Kingdom, March 2-6, 2024*, pages 348–363. ACM, 2024.

4.3 Towards a theory of tile-centric computation

Johannes Langguth (Simula Research Laboratory – Oslo, NO)

License © Creative Commons BY 4.0 International license
© Johannes Langguth

The first and foremost topic of the working group was to define tile-centric computing (TCC) and tile-centric architectures (TCAs). It is widely agreed that TCAs consist of a large number of relatively small cores which directly connected to SRAM that is used as memory. Together, a cores with its SRAM is referred to as a tile. There are some instances of such TCAs in use today; these include the Graphcore Intelligence Processing Unit (IPU) and the Cerebras Systems Wafer-Scale Engine (WSE).

This implies a focus on MIMD processing, although many candidate architectures also have SIMD capabilities. TCAs thus differ from GPUs which strongly rely on wide SIMD and typically contain small amounts of SRAM. They also differ from CPUs, despite the fact that modern CPUs often contain a large number of cores connected to SRAM. The crucial difference is the use of SRAM as user-controlled memory, rather than cache that buffers accesses to larger DRAM or HBM in the case of CPUs.

The definition of interconnects between tiles made for a more lively discussion, especially because the IPU and WSE differ substantially in this regard. While communication between any pair of tiles on the IPU is almost equal, the 2D interconnect of the WSE makes physical location in the tile grid very important. Furthermore, devices from several other vendors, including SambaNova, Tenstorrent, and Groq, have a grid structure that is somewhat similar

to the WSE, although it is not yet clear which of these architectures are suitable for graph algorithms. Thus, the consensus was that data location in the tile grid is a crucial part of TCC and that the IPU is an outlier in this regard. In any case, previous work has shown that when using multiple IPUs, the locality problem again becomes highly relevant.

The accepted term for this idea is spatial computing, but since Apple is currently using the term for its augmented reality product, some felt that the use of the term in a general discussion is discouraged to avoid confusion. While this is similar to standard distributed memory computation, it is important to stress that the tile-centric view considers the computation to be shared among the tiles, rather than composed of independent computations on e.g. MPI ranks, as is the case in message passing. While this distinction may sound overly fine, an important consequence is that for tile-centric computation, different groups of tiles having completely different functions is the norm rather than the exception.

Having defined, loosely, the class of TCAs and shared an understanding of the specifics of some instances, the discussion focused on the implications: what questions do TCAs raise for the graph algorithms community. There was a wide agreement that some sort of abstraction layer such as graphBLAS is needed since the low level implementation on TCAs clearly seems more difficult than on CPUs and likely also on GPUs. W.r.t. memory, a SHMEM or PGAS-like interface would be desirable. In addition, there is a need for adapting existing partitioning algorithms to the requirements of tile-centric devices. The 2D interconnect of the WSE also calls for algorithms that embed graphs into 2D space. Finally it was agreed that advanced and unconventional algorithmic concepts such as temporal data sharing are worth investigating on the new devices, although nothing concrete has been discussed so far

4.4 Adjoint Differentiation and Its Challenges

Johannes Lotz (NAG – Oxford, GB), Martin Bucker (Friedrich-Schiller-Universität Jena, DE), and Paul D. Hovland (Argonne National Laboratory, US)


License © Creative Commons BY 4.0 International license
© Johannes Lotz, Martin Bucker, and Paul D. Hovland

Adjoints of arbitrary differentiable programs can be computed by an algorithm similar to backpropagation in artificial neural networks. They are crucial ingredients of the CSE toolbox. A major obstacle for an efficient implementation is the need for reversal of the data flow, which yields a number of hard combinatorial optimization problems.

We were a group of six individuals. Although we had a prearranged list of topics, we commenced by brainstorming the most engaging and promising subjects for the group. Ultimately, we delved deeply into two main topics: [A] The use of Automatic Adjoint Differentiation (AAD) in finance, focusing on its hardware implications and the challenges associated with the Partial Differential Equations (PDE) approach to Stochastic Differential Equations (SDEs). And [B], the propagation of compressed Jacobians through a chain of sparse Jacobians. Our discussions on both topics were productive and led to the following outcomes: For [A], a sub-group consisting of Uwe Naumann, Johannes Lotz, and Jason Charlesworth decided to schedule a follow-up meeting. And for [B], despite the initial promise of the idea, the group thoroughly analyzed it and concluded that it did not contain any further significant potential.

4.5 Hypergraph Algorithms Working Group

Nate Veldt (Texas A&M University – College Station, US), Alex Crane (University of Utah – Salt Lake City, US), Gero Kauerauf (RWTH Aachen, DE), Daniel Král’ (Masaryk University – Brno, CZ), Henning Meyerhenke (HU Berlin, DE), Henrik Reinstädler (Universität Heidelberg, DE), Christian Schulz (Universität Heidelberg, DE), Blair D. Sullivan (University of Utah – Salt Lake City, US), and Bora Uçar (ENS – Lyon, FR)

License  Creative Commons BY 4.0 International license
© Nate Veldt, Alex Crane, Gero Kauerauf, Daniel Král’, Henning Meyerhenke, Henrik Reinstädler, Christian Schulz, Blair D. Sullivan, and Bora Uçar

Hypergraphs generalize graphs by allowing edges (also called hyperedges) to include an arbitrary number of nodes, rather than just two. Hypergraph representations and algorithms have been used in scientific computing applications for decades, and have recently have been growing in popularity within the machine learning and data mining communities.

The working group on hypergraph algorithms specifically explored various extensions and algorithms for a clustering framework called edge colored clustering (ECC) [1, 2]. The input to the problem is a hypergraph in which every edge is associated with a color, and the goal is to assign colors to nodes in order to maximize the number of satisfied edges, where a satisfied edge is one in which all nodes within the edge are assigned the same color as the edge. This framework has been used as a model for clustering objects based on the group interactions they participate in (the hyperedges) as well as the *type* or *category* of interaction (represented by the hyperedge color).

During our working group discussions we made progress on several variants and extensions of the ECC problem. This includes (1) identifying new connections between a variant of ECC with overlapping clusters and the concept of b-matchings in hypergraphs; (2) developing a new streaming model for the problem and providing an initial analysis of the tradeoffs between space and number of passes in designing approximation algorithms; and (3) identifying key challenges in extending parameterized algorithms for the graph version of the problem to the hypergraph setting. For the latter direction, we designed a polynomial kernel for checking whether t hyperedges can be satisfied in a given hypergraph, whose size depends explicitly on the rank r (the maximum hyperedge size).

References

- 1 Alex Crane, Brian Lavallee, Blair D. Sullivan, and Nate Veldt, *Overlapping and robust edge-colored clustering in hypergraphs*, Proceedings of the 17th ACM International Conference on Web Search and Data Mining, pages 143–151, 2024.
- 2 Nate Veldt, *Optimal LP rounding and linear-time approximation algorithms for clustering edge-colored hypergraphs*, International Conference on Machine Learning, pages 34924–34951, 2023. PMLR.

5 Panel discussions

5.1 Algorithms: Beyond the Static

Kathrin Hanauer (Universität Wien, AT), Quanquan C. Liu (Yale University – New Haven, US), Manuel Penschuck (Goethe University – Frankfurt am Main, DE), and Helen Xu (Georgia Institute of Technology – Atlanta, US)

License © Creative Commons BY 4.0 International license
© Kathrin Hanauer, Quanquan C. Liu, Manuel Penschuck, and Helen Xu

In the past, algorithms were often viewed as static entities, optimized for specific problems and processing constraints. However, with the explosion of big data, the advent of modern CPUs and the broad availability of computing clusters, the algorithm landscape has undergone a profound shift. Today, we see – among others – a growing demand for algorithms that can adapt dynamically to changing inputs, leverage parallel processing, harness the power of distributed computing, and boost performance by integrating techniques from machine learning.

In this session, we gave an introduction and overview over these modern algorithms and discussed both theoretical advancements and practical applications as part of the following talks:

- Dynamic Graph Algorithms in Theory and Practice (Kathrin Hanauer, Universität Wien, AT; [1])
- Developing and Benchmarking Large-Scale Dynamic-Graph Containers (Helen Xu, Georgia Institute of Technology – Atlanta, US)
- Sampling Practical and Scalable Sampling Algorithms (Manuel Penschuck, Goethe University – Frankfurt am Main, DE)
- Learning-Augmented Algorithms (Quanquan C. Liu, Yale University – New Haven, US)

References

- 1 Kathrin Hanauer, Monika Henzinger, and Christian Schulz. Fully dynamic single-source reachability in practice: An experimental study. In Guy E. Blelloch and Irene Finocchi, editors, *Proceedings of the Symposium on Algorithm Engineering and Experiments, ALENEX 2020, Salt Lake City, UT, USA, January 5-6, 2020*, pages 106–119. SIAM, 2020.

5.2 The Future of Computing

Bruce Hendrickson (LLNL – Livermore, US), Jakob Engblom (Intel Sweden – Kista, SE), Chris Goodyer (Arm – Manchester, GB), and Oded Green (NVIDIA – Alpharetta, US)

License © Creative Commons BY 4.0 International license
© Bruce Hendrickson, Jakob Engblom, Chris Goodyer, and Oded Green

A combination of technological and market forces are driving rapid changes in computer architectures and system designs. These changes will have significant impacts for algorithms and software. This session reviewed the drivers behind these changes and provided some thoughts on what the future might look like. The goal of this session was to provide context for much of the remainder of the Dagstuhl program.

The session involved four presentations and lots of discussion. Bruce Hendrickson of Lawrence Livermore National Lab moderated the session and provided an overview / introductory talk. This was followed by presentations from employees of three major

computing companies sharing their thoughts on potential future paths. Chris Goodyer of ARM spoke on “Building the future of computing.” Oded Green of Nvidia talked about “The Future of Accelerated Combinatorial and Sparse Applications.” And Jakob Engblom of Intel covered “Just Add Accelerators – The Answer to Everything?”

5.3 MLQAOA: Graph Learning Accelerated Hybrid Quantum-Classical Multilevel QAOA

Ilya Safro (University of Delaware – Newark, US) and Eleanor Rieffel (NASA – Moffett Field, US)

License © Creative Commons BY 4.0 International license
© Ilya Safro and Eleanor Rieffel

Main reference Bao Bach, Jose Falla, Ilya Safro: “MLQAOA: Graph Learning Accelerated Hybrid Quantum-Classical Multilevel QAOA”, CoRR, Vol. abs/2404.14399, 2024.

URL <http://dx.doi.org/10.48550/ARXIV.2404.14399>

I. Safro: This is an introductory talk on the basics of quantum computing. We will introduce qubits, quantum gates, circuits, basic principles of quantum mechanics for computing, entanglement and several algorithms including Bernstein-Vazirani, Simon’s problem, Shor’s algorithm, Hidden Subgroup Problem and Grover search.

I. Safro: Learning the problem structure at multiple levels of coarseness to inform the decomposition-based hybrid quantum-classical combinatorial optimization solvers is a promising approach to scaling up variational approaches. We introduce a multilevel algorithm reinforced with the spectral graph representation learning-based accelerator to tackle large-scale graph maximum cut instances and fused with several versions of the quantum approximate optimization algorithm (QAOA) and QAOA-inspired algorithms. The graph representation learning model utilizes the idea of QAOA variational parameters concentration and substantially improves the performance of QAOA. We demonstrate the potential of using multilevel QAOA and representation learning-based approaches on very large graphs by achieving high-quality solutions in a much faster time. This talk is based on several recent works [1],[2],[3], and [4].

E. Rieffel: The talk begins with an overview of the status of current quantum processors, followed by a vision of future quantum computers. We will discuss commonalities and differences between application-scale fault-tolerant quantum computing architectures (which will necessarily contain many networked quantum and classical (non-quantum) processors) and supercomputer architectures. We will then discuss the status of quantum algorithms generally, before focusing on quantum and hybrid quantum-classical optimization algorithms, with a mention of ties to sampling and machine learning. We then highlight quantum-inspired classical algorithms and hardware. The last part of the talk gives brief glimpses of other topics with relation to discrete problems and algorithms, including compilation of algorithms to quantum hardware, quantum error correction, and polytopes arising in fundamental quantum mechanics and quantum information theory.

References

- 1 Bao Bach, Jose Falla, Ilya Safro *MLQAOA: Graph Learning Accelerated Hybrid Quantum-Classical Multilevel QAOA*. submitted, arXiv:2404.14399, 2024
- 2 Anthony Angone, Xioayuan Liu, Ruslan Shaydulin, Ilya Safro *Hybrid Quantum-Classical Multilevel Approach for Maximum Cuts on Graphs*. IEEE High-Performance and Extreme Computing (HPEC), preprint at <https://arxiv.org/pdf/2309.08815.pdf>, 2023

- 3 Hayato Ushijima-Mwesigwa, Ruslan Shaydulin, Susan Mniszewski, Christian Negre, Yuri Alexeev, Ilya Safro *Multilevel Combinatorial Optimization Across Quantum Architectures*, ACM Transactions on Quantum Computing, Vol. 2(1), pp. 1-29, 2021, preprint at <https://arxiv.org/abs/1910.09985>
- 4 Jose Falla, Quinn Langfitt, Yuri Alexeev, Ilya Safro *Graph Representation Learning for Parameter Transferability in Quantum Approximate Optimization Algorithm*, submitted, preprint at <https://arxiv.org/abs/2401.06655>, 2024

5.4 Wafer Scale Computing: Fine Grain Parallelism and Rethinking Parallel Computing

Robert Schreiber (Cerebras Systems – Palo Alto, US)

License © Creative Commons BY 4.0 International license
© Robert Schreiber
Joint work of The whole Cerebras engineering team
URL www.cerebras.net

I will explain how wafer-scale computing currently works by detailing the hardware, architecture, and programming paradigms of the Cerebras machines, the only instance of commercial wafer-scale computers today.

The CS-3 incorporates all memory and processing on one wafer, a wafer that contains 900,000 processing elements. With 48KB of local memory, a PE cannot hold very much data. On the other hand, access to that data is at the same rate as peak speed computation. Most interesting, the mesh interconnect has single-clock latency for sending a message (of 4 bytes) to a mesh neighboring PE, and the network can sustain a 4 byte message to and from each neighbor on every clock.

The wafer is therefore a working instance of processing co-located with memory. While it is distributed memory from the addressing perspective, the extreme interconnect performance allows programmers to treat distributed tensors as if they were shared – shared objects in a distributed memory substrate. This finds uses in graph computing, sparse matrix vector products, neutron transport applications, for some examples.

The absence of both memory walls and slow, high-overhead, high-latency interconnect permits very fine grained parallel applications that achieve excellent performance. This in turn allows strong scaling in which each PE holds only a few words of the problem data, taking full advantage of the easy accessibility of data on near neighbor PEs. Thus, strong scaling is quite successful, which reduces runtimes for problems of the scale that fit the wafer by two orders of magnitude, allowing applications that are impossible with conventional systems. I will cover some use cases and give an outline of how the system can be programmed using the Cerebras SDK.

Participants

- Cevdet Aykanat
Bilkent University – Ankara, TR
- David A. Bader
NJIT – Newark, US
- Paolo Bientinesi
University of Umeå, SE
- Erik Boman
Sandia National Labs –
Albuquerque, US
- Martin Bücker
Friedrich-Schiller-Universität
Jena, DE
- Jason Charlesworth
Zettamatics – Manningtree, GB
- Cedric Chevalier
CEA – Bruyères-le-Châtel, FR
- Alex Crane
University of Utah –
Salt Lake City, US
- Jakob Engblom
Intel Sweden – Kista, SE
- John Gilbert
University of California –
Santa Barbara, US
- Chris Goodyer
Arm – Manchester, GB
- Oded Green
NVIDIA – Alpharetta, US
- Inge Li Gørtz
Technical University of Denmark
– Lyngby, DK
- Kathrin Hanauer
Universität Wien, AT
- Bruce Hendrickson
LLNL – Livermore, US
- Paul D. Hovland
Argonne National Laboratory, US
- Nidhi Kaihnsa
University of Copenhagen, DK
- Gero Kauer auf
RWTH Aachen, DE
- Neil Kichler
RWTH Aachen, DE
- Daniel Král’
Masaryk University – Brno, CZ
- Johannes Langguth
Simula Research Laboratory –
Oslo, NO
- Klaus Leppkes
Ex-Luminary Cloud –
Redwood City, US
- Quanquan C. Liu
Yale University – New Haven, US
- Johannes Lotz
NAG – Oxford, GB
- Fredrik Manne
University of Bergen, NO
- Henning Meyerhenke
HU Berlin, DE
- Uwe Naumann
RWTH Aachen, DE
- Manuel Penschuck
Goethe University –
Frankfurt am Main, DE
- Alex Pothén
Purdue University – West
Lafayette, US
- Arash Pourdamghani
TU Berlin, DE
- Henrik Reinstädler
Universität Heidelberg, DE
- Eleanor Rieffel
NASA – Moffett Field, US
- Ilya Safro
University of Delaware –
Newark, US
- Robert Schreiber
Cerebras Systems –
Palo Alto, US
- Christian Schulz
Universität Heidelberg, DE
- Angelika Schwarz
Sona, SE
- George Slota
Rensselaer Polytechnic Institute –
Troy, US
- Sergii Strelchuk
University of Warwick –
Coventry, GB
- Blair D. Sullivan
University of Utah –
Salt Lake City, US
- Sivan Toledo
Tel Aviv University, IL
- Bora Uçar
ENS – Lyon, FR
- Nate Veldt
Texas A&M University –
College Station, US
- Helen Xu
Georgia Institute of Technology –
Atlanta, US

