Reading list for CS 214: Parallel Algorithms

YAN GU, University of California, Riverside

The following is the Reading list for CS 214: Parallel Algorithms. This reading list includes recent advances for parallel algorithms, categorized in various topics. You should read and select some topics for paper reading and the course presentation from the this list.

We have covered many algorithms during the lectures, but as a 10-week class, we do not have time to cover many algorithms for important problems, and ideas for parallel algorithm design. As a result, we provide this list of papers with brief introduction. You can read these papers to enhance your understanding of the course content, and pick topics for the optional course presentations in the last week. You can either give an overview talk for the high-level ideas in several papers, focus on one specific paper, or just introduce one algorithm in one paper (but with more depth).

If you want to give this talk, make sure you contact the instructor regarding the content, and make sure the length fits in the time slots.

If the slots for course presentation is full, you can alternatively submit paper reading for bonus credits. A paper reading should contains 1000–3000 words, and will be graded by the quality of the writing.

1 GENERAL IDEAS FOR PARALLEL ALGORITHMS

In class, we mainly focus on the binary fork-join model. We mainly talk about the simplest algorithm for each problem, and mention in several places that the bounds can be improved by some recent papers. Many of these results are from the follow paper: "Optimal Parallel Algorithms in the Binary-Forking Model" [11]. These algorithms are quite simple, and you are welcome to discuss some of them.

Space consumption is another critical objective we should take care of when designing parallel algorithms, as they are supposed to process large-scale data. A parallel algorithm that are space-efficient can run larger instances on the same machine, and is presumably faster since they have smaller memory footprint. Many interesting space-efficient parallel algorithms are designed in this paper: "Parallel In-Place Algorithms: Theory and Practice" [35]. You are welcome to talk about the high-level idea in designing these algorithms, and provide some specific examples.

Given a graph with a certain computational DAG, are there some general approaches to parallelize this computation? We refer you to the following two papers: "Internally Deterministic Parallel Algorithms Can Be Fast" [10] and "Many Sequential Iterative Algorithms Can Be Parallel and (Nearly) Work-efficient" [42] for some interesting discussions.

2 SEQUENCE ALGORITHMS

Sequence algorithms, which are algorithms working on 1D static data such as sorting, are crucial primitives for general algorithm design. You are welcome to read the following papers and discuss some of them in details. These papers include: comparison sort (sample sort) [4, 13], integer sort [41], and semisort [27, 36]; prefix sums [5]; random permutation, list ranking, and tree contraction [11, 35, 48].

Author's address: Yan Gu, University of California, Riverside.

3 DATA STRUCTURES

In class we talk about parallel data structure design, and the advantages over the classic concurrent data structures. We use hash table and binary trees as examples, and we welcome some in-depth discussion and/or analysis for these algorithms. These papers include: hash tables [44]; binary search trees [8, 9, 53], range, interval and segment trees [51], and persistent trees [52]; priority queue [25]; Cartesian trees [45] and suffix arrays [38]; cover trees [34]; van Emde Boas trees [32]; union-find [50]; and rake-compress trees [1].

4 GRAPHS ANALYTICS

Graph analytics is a crucial component in algorithm design and parallel algorithm research. Research has span across programming-level abstraction, specific algorithms, and support for dynamic updates, compression, etc.

Regarding graph-processing systems, there has been extensive study on this topic. Here we only list a small number of them, but you are welcome to read and present works not on this list. Ligra (A Lightweight Graph Processing Framework for Shared Memory [43]) is one of the earlist systems focusing on abstracting and optimizing BFS-like round-based traversal algorithms. Julienne (A Framework for Parallel Graph Algorithms using Work-efficient Bucketing [21]) extended Ligra to accommodate applications such as shortest-paths and k-core in an efficient way. GraphChi (Large-Scale Graph Computation on Just a PC [39]) is an early and influential disk-based work, and SAGE [6, 24] focus on non-volatile main memories (NVRAMs).

Parallel graph algorithms are notoriously hard to design, but over the decades, many simple and elegant algorithms have been shown. We will list the state-of-the-art algorithms for famous problems. Graph connectivity is extensively studied by these two papers [23, 46]. The latest biconnectivity algorithms, the FAST-BCC algorithm, is given in [26]. On directed graph, the problem is refer to strong connectivity. The latest algorithm, the BGSS algorithm, is given in [11], and the implementation is in [55]. The state-of-the-art shortest path algorithms are given in [25], inspired by earlier work [18, 40]. Other interesting graph algorithms include: k-core [21], maximal independent set, maximal matching, and graph coloring [12, 42].

Finally, it has been a long history on dealing with dynamic evolving graphs. Early work includes STINGER [28], among many others. A recent breakthrough is to use dynamic trees to represent the edge lists of the CSR format, which supports efficient updates, multi-versioning, and hybrid update and query. This series of work include the first version, Aspen [22], and the state-of-the-art version, PaC-tree [20]. Both solutions rely on graph compression techniques, and more discussions can be found in [47].

5 GEOMETRY PROCESSING

While graphs study the relationship among objects, geometry directly studies the locations among objects. There have been many recent advances for parallel geometry processing, and we provide a list of references if you are interested in discussing.

The first type of work is data structures that organizes data in low-to-mid dimension space. Aside from range, interval and segment trees [51] and cover trees [34] mentioned above, other work of interest includes bounding volume hierarchies [31, 54] and k-d trees [7, 15].

The second groups of work are parallel algorithms for classic geometric problems. A recent work by Blelloch et al. [16] shows parallel algorithms for a long list of problems such as Delaunay triangulation, low-dimension LP, closest pairs [58], etc. The latest convex hull paper is [17], and the classic divide-and-conquer approach is [30].

The last group of work contains geometric algorithms and highly optimized implantations for data clustering, and some papers of interest include: DBSCAN [56], Euclidean MST and hierarchical clustering [57], and agglomerative clustering [59].

6 INCREMENTAL CONSTRUCTION

A significant recent advance for parallel algorithms is on how to directly parallelize sequential algorithms. A list of papers on this direction includes (may overlap the above ones): maximal independent set and maximal matching [12, 29, 42]; graph coloring [37]; correlation clustering [3]; random permutation, list contraction, and tree contraction [11, 49]; comparison sorting, Delaunay triangulation, linear programming, smallest enclosing disk, and closest pairs [15, 16], convex hull [17]; cover tree [34]; and longest increasing subsequences (LIS) and various DP algorithms [14, 42].

7 SCHEDULING

Scheduling plays a crucial part in today's parallel programming world. It decouples the low-level details (e.g., task distribution, dynamic load balancing) from the high-level algorithmic insights (i.e., what can be parallelized in a computation). Most existing software/libraries uses work-stealing schedulers in practice, which are not only fast in practice, but have strong theoretical guarantees. A list of classic papers for the analysis of the schedulers are: [2, 19, 33].

REFERENCES

[1] Umut A Acar, Daniel Anderson, Guy E Blelloch, and Laxman Dhulipala. Parallel batch-dynamic graph connectivity. In *ACM Symposium on Parallelism in Algorithms and Architectures*, pages 381–392, 2019.

[2] Umut A. Acar, Guy E. Blelloch, and Robert D. Blumofe. The data locality of work stealing. *Theoretical Computer Science (TCS)*, 35(3), 2002.

[3] KookJin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. Correlation clustering in data streams. In *International Conference on Machine Learning (ICML)*, pages 2237–2246, 2015.

[4] Michael Axtmann, Sascha Witt, Daniel Ferizovic, and Peter Sanders. In-place parallel super scalar samplesort (ipsssso). In *European Symposium on Algorithms (ESA)*, 2017.

[5] Guy E. Blelloch. Prefix sums and their applications. In John Reif, editor, *Synthesis of Parallel Algorithms*. Morgan Kaufmann, 1993.

[6] Guy E Blelloch, Laxman Dhulipala, Phillip B Gibbons, Yan Gu, Charles McGuffey, and Julian Shun. The read-only semi-external model. In *SIAM Symposium on Algorithmic Principles of Computer Systems (APOCS)*, pages 70–84. SIAM, 2021.

[7] Guy E Blelloch and Magdalen Dobson. Parallel nearest neighbors in low dimensions with batch updates. In *Algorithm Engineering and Experiments (ALENEX)*, pages 195–208. SIAM, 2022.

[8] Guy E. Blelloch, Daniel Ferizovic, and Yihan Sun. Just join for parallel ordered sets. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2016.

[9] Guy E. Blelloch, Daniel Ferizovic, and Yihan Sun. Joinable parallel balanced binary trees. In *ACM Transactions on Parallel Computing (TOPC)*, 2022.

[10] Guy E. Blelloch, Jeremy T. Fineman, Phillip B. Gibbons, and Julian Shun. Internally deterministic parallel algorithms can be fast. In *ACM Symposium on Principles and Practice of Parallel Programming (PPOPP)*, 2012.

[11] Guy E. Blelloch, Jeremy T. Fineman, Yan Gu, and Yihan Sun. Optimal parallel algorithms in the binary-forking model. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2020.

[12] Guy E. Blelloch, Jeremy T Fineman, and Julian Shun. Greedy sequential maximal independent set and matching are parallel on average. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2012.

[13] Guy E. Blelloch, Phillip B. Gibbons, and Harsha Vardhan Simhadri. Low depth cache-oblivious algorithms. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2010.

[14] Guy E. Blelloch and Yan Gu. Improved parallel cache-oblivious algorithms for dynamic programming. In *SIAM Symposium on Algorithmic Principles of Computer Systems (APOCS)*, 2020.

[15] Guy E. Blelloch, Yan Gu, Julian Shun, and Yihan Sun. Parallel write-efficient algorithms and data structures for computational geometry. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2018.

[16] Guy E. Blelloch, Yan Gu, Julian Shun, and Yihan Sun. Parallelism in randomized incremental algorithms. *J. ACM*, 2020.

[17] Guy E. Blelloch, Yan Gu, Julian Shun, and Yihan Sun. Randomized incremental convex hull is highly parallel. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2020.

[18] Guy E. Blelloch, Yan Gu, Yihan Sun, and Kanat Tangwongsan. Parallel shortest paths using radius stepping. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2016.

[19] Robert D. Blumofe and Charles E. Leiserson. Scheduling multithreaded computations by work stealing. *J. ACM*, 46(5):720–748, 1999.

[20] Laxman Dhulipala, Guy Blelloch, Yan Gu, and Yihan Sun. Pac-trees: Supporting parallel and compressed purely-functional collections. In *manuscript*, 2022.

[21] Laxman Dhulipala, Guy E. Blelloch, and Julian Shun. Julienne: A framework for parallel graph algorithms using work-efficient bucketing. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2017.

[22] Laxman Dhulipala, Guy E Blelloch, and Julian Shun. Low-latency graph streaming using compressed purely-functional trees. In *ACM Conference on Programming Language Design and Implementation (PLDI)*, pages 918–934, 2019.

[23] Laxman Dhulipala, Changwan Hong, and Julian Shun. Connectit: a framework for static and incremental parallel graph connectivity algorithms. *Proceedings of the VLDB Endowment (PVLDB)*, 14(4):653–667, 2020.

[24] Laxman Dhulipala, Charlie McGuffey, Hongbo Kang, Yan Gu, Guy E Blelloch, Phillip B Gibbons, and Julian Shun. Semi-asymmetric parallel graph algorithms for NVRAMs. *Proceedings of the VLDB Endowment (PVLDB)*, 13(9), 2020.

[25] Xiaojun Dong, Yan Gu, Yihan Sun, and Yunming Zhang. Efficient stepping algorithms and implementations for parallel shortest paths. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2021.

[26] Xiaojun Dong, Letong Wang, Yan Gu, and Yihan Sun. Provably fast and space-efficient parallel biconnectivity. *ACM Symposium on Principles and Practice of Parallel Programming (PPOPP)*, 2023.

[27] Xiaojun Dong, Yunshu Wu, Zhongqi Wang, Laxman Dhulipala, Yan Gu, and Yihan Sun. High-performance and flexible parallel algorithms for semisort and related problems. In *manuscript*, 2023.

[28] David Ediger, Robert McColl, Jason Riedy, and David A Bader. Stinger: High performance data structure for streaming graphs. In *IEEE Conference on High Performance Extreme Computing (HPEC)*, pages 1–5, 2012.

[29] Manuela Fischer and Andreas Noever. Tight analysis of parallel randomized greedy mis. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2152–2160, 2018.

[30] Michael T Goodrich. Finding the convex hull of a sorted point set in parallel. *Information Processing Letters*, 26(4), 1987.

[31] Yan Gu, Yong He, Kayvon Fatahalian, and Guy Blelloch. Efficient BVH construction via approximate agglomerative clustering. In *High-Performance Graphics (HPG)*, 2013.

[32] Yan Gu, Ziyang Men, Zheqi Shen, Yihan Sun, and Zijin Wan. Parallel longest increasing subsequence and van emde boas trees. In *manuscript*, 2023.

[33] Yan Gu, Zachary Napier, and Yihan Sun. Analysis of work-stealing and parallel cache complexity. In *SIAM Symposium on Algorithmic Principles of Computer Systems (APOCS)*, pages 46–60. SIAM, 2022.

[34] Yan Gu, Zachary Napier, Yihan Sun, and Letong Wang. Parallel cover trees and their applications. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 259–272, 2022.

[35] Yan Gu, Omar Obeya, and Julian Shun. Parallel in-place algorithms: Theory and practice. In *SIAM Symposium on Algorithmic Principles of Computer Systems (APOCS)*, pages 114–128, 2021.

[36] Yan Gu, Julian Shun, Yihan Sun, and Guy E. Blelloch. A top-down parallel semisort. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 24–34, 2015.

[37] William Hasenplaugh, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. Ordering heuristics for parallel graph coloring. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2014.

[38] Juha Kärkkäinen and Peter Sanders. Simple linear work suffix array construction. In *Intl. Colloq. on Automata, Languages and Programming (ICALP)*, pages 943–955. Springer, 2003.

[39] Aapo Kyrola, Guy E. Blelloch, and Carlos Guestrin. Graphchi: Large-scale graph computation on just a PC. In *USENIX conference on Operating Systems Design and Implementation (OSDI)*, 2012.

[40] Ulrich Meyer and Peter Sanders. Δ-stepping: a parallelizable shortest path algorithm. *Journal of Algorithms*, 49(1):114–152, 2003.

[41] Omar Obeya, Endrias Kahssay, Edward Fan, and Julian Shun. Theoretically-efficient and practical parallel in-place radix sorting. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 213–224, 2019.

[42] Zheqi Shen, Zijin Wan, Yan Gu, and Yihan Sun. Many sequential iterative algorithms can be parallel and (nearly) work-efficient. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2022.

[43] Julian Shun and Guy E. Blelloch. Ligra: A lightweight graph processing framework for shared memory. In *ACM Symposium on Principles and Practice of Parallel Programming (PPOPP)*, 2013.

[44] Julian Shun and Guy E Blelloch. Phase-concurrent hash tables for determinism. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 96–107, 2014.

[45] Julian Shun and Guy E. Blelloch. A simple parallel cartesian tree algorithm and its application to parallel suffix tree construction. *ACM Transactions on Parallel Computing (TOPC)*, 1(1), October 2014.

[46] Julian Shun, Laxman Dhulipala, and Guy Blelloch. A simple and practical linear-work parallel algorithm for connectivity. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2014.

[47] Julian Shun, Laxman Dhulipala, and Guy E Blelloch. Smaller and faster: Parallel processing of compressed graphs with ligra+. In *IEEE Data Compression Conference (DCC)*, pages 403–412, 2015.

[48] Julian Shun, Yan Gu, Guy E. Blelloch, Jeremy T Fineman, and Phillip B Gibbons. Sequential random permutation, list contraction and tree contraction are highly parallel. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 431–448, 2015.

[49] Julian Shun, Yan Gu, Guy E. Blelloch, Jeremy T. Fineman, and Phillip B. Gibbons. Sequential random permutation, list contraction and tree contraction are highly parallel. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 431–448, 2015.

[50] Natcha Simsiri, Kanat Tangwongsan, Srikanta Tirthapura, and Kun-Lung Wu. Work-efficient parallel union-find with applications to incremental graph connectivity. In *European Conference on Parallel Processing (Euro-Par)*, 2016.

[51] Yihan Sun and Guy E Blelloch. Parallel range, segment and rectangle queries with augmented maps. In *SIAM Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 159–173, 2019.

[52] Yihan Sun, Guy E Blelloch, Wan Shen Lim, and Andrew Pavlo. On supporting efficient snapshot isolation for hybrid workloads with multi-versioned indexes. *Proceedings of the VLDB Endowment (PVLDB)*, 13(2):211–225, 2019.

[53] Yihan Sun, Daniel Ferizovic, and Guy E Blelloch. Pam: Parallel augmented maps. In *ACM Symposium on Principles and Practice of Parallel Programming (PPOPP)*, 2018.

[54] Ingo Wald, Thiago Ize, and Steven G Parker. Fast, parallel, and asynchronous construction of bvhs for ray tracing animated scenes. *Computers & Graphics*, 32(1):3–13, 2008.

[55] Letong Wang, Xiaojun Dong, Yan Gu, and Yihan Sun. Parallel strong connectivity based on faster reachability. 2023.

[56] Yiqiu Wang, Yan Gu, and Julian Shun. Theoretically-efficient and practical parallel dbscan. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 2555–2571, 2020.

[57] Yiqiu Wang, Shangdi Yu, Yan Gu, and Julian Shun. Fast parallel algorithms for euclidean minimum spanning tree and hierarchical spatial clustering. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 1982–1995, 2021.

[58] Yiqiu Wang, Shangdi Yu, Yan Gu, and Julian Shun. A parallel batch-dynamic data structure for the closest pair problem. In *ACM Symposium on Computational Geometry (SoCG)*, 2021.

[59] Shangdi Yu, Yiqiu Wang, Yan Gu, Laxman Dhulipala, and Julian Shun. Parchain: A framework for parallel hierarchical agglomerative clustering using nearest-neighbor chain. *Proceedings of the VLDB Endowment (PVLDB)*, 2021.