Massively Parallel Computation: Algorithms and Applications

Other titles in Foundations and Trends® in Optimization

Acceleration Methods
Alexandre d'Aspremont, Damien Scieur and Adrien Taylor
ISBN: 978-1-68083-928-9

Algorithms for Verifying Deep Neural Networks
Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher Strong, Clark Barrett and Mykel J. Kochenderfer
ISBN: 978-1-68083-786-5

Distributionally Robust Learning
Ruidi Chen and Ioannis Ch. Paschalidis
ISBN: 978-1-68083-772-8

Atomic Decomposition via Polar Alignment: The Geometry of Structured Optimization
Zhenan Fan, Halyun Jeong, Yifan Sun and Michael P. Friedlander
ISBN: 978-1-68083-742-1

Optimization Methods for Financial Index Tracking: From Theory to Practice
Konstantinos Benidis, Yiyong Feng and Daniel P. Palomar
ISBN: 978-1-68083-464-2

Massively Parallel Computation: Algorithms and Applications

Sungjin Im
University of California, Merced

Ravi Kumar
Google, Mountain View

Silvio Lattanzi
Google, Barcelona

Benjamin Moseley
Carnegie Mellon University

Sergei Vassilvitskii
Google, New York

now
the essence of knowledge
Boston — Delft

Foundations and Trends® in Optimization

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

S. Im *et al.*. *Massively Parallel Computation: Algorithms and Applications*. Foundations and Trends® in Optimization, vol. 5, no. 4, pp. 340–417, 2023.

ISBN: 978-1-63828-217-4

Foundations and Trends® in Optimization
Volume 5, Issue 4, 2023

Editorial Scope

Topics

Foundations and Trends® in Optimization publishes survey and tutorial articles in the following topics:

- algorithm design, analysis, and implementation (especially, on modern computing platforms
- models and modeling systems, new optimization formulations for practical problems
- applications of optimization in machine learning, statistics, and data analysis, signal and image processing, computational economics and finance, engineering design, scheduling and resource allocation, and other areas

Information for Librarians

Foundations and Trends® in Optimization, 2023, Volume 5, 4 issues. ISSN paper version 2167-3888. ISSN online version 2167-3918. Also available as a combined paper and online subscription.

Contents

1 Introduction **2**

1.1 Purpose of This Monograph 4

1.2 Prerequisites 5

2 The MPC Model **6**

2.1 Formal Definition 7

2.2 Example: Word Frequencies in Two Rounds 9

2.3 Other Related Models 11

2.4 Section Notes 14

3 Partitioning and Coresets **15**

3.1 Overview 16

3.2 Application: Minimum Spanning Tree 16

3.3 Application: k-Center Clustering 18

3.4 Coresets 19

3.5 Application: k-Center Clustering in Euclidean Space 20

3.6 Problems 21

3.7 Section Notes 22

4 Sample and Prune **23**

4.1 Overview 23

4.2 Application: Top k Selection 27

4.3 Application: k-Center Clustering 27
4.4 Application: Monotone Submodular Maximization Subject to a Cardinality Constraint 29
4.5 Section Notes 30

5 Dynamic Programming **32**
5.1 Overview 33
5.2 Warm-up: Knapsack 35
5.3 Interval Selection in MPC 37
5.4 Approximate Dynamic Programs 40
5.5 Section Notes 41

6 Round Reduction via Sampling **42**
6.1 k-core Decomposition and a Sequential Algorithm 42
6.2 Parallelizing the Sequential Algorithm: $O(\log n)$ Rounds . . 44
6.3 Round Compression via Random Vertex Partitioning: $O(\log \log n)$ Rounds 47
6.4 Section Notes 49

7 Round Reduction via Graph Exponentiation **51**
7.1 Approximate Core Decomposition 52
7.2 Connected Components 57
7.3 Section Notes 63

8 Lower Bounds **64**
8.1 Connectivity in MPC 64
8.2 Unconditional Lower Bounds 65
8.3 Conditional Lower Bounds 69
8.4 Section Notes 71

9 Conclusions **72**

References **73**

Massively Parallel Computation: Algorithms and Applications

Sungjin Im[1], Ravi Kumar[2], Silvio Lattanzi[3], Benjamin Moseley[4] and Sergei Vassilvitskii[5]

[1] *University of California, Merced, USA; sim3@ucmerced.edu*
[2] *Google, Mountain View, USA; ravi.k53@gmail.com*
[3] *Google, Barcelona, Spain; silviol@google.com*
[4] *Carnegie Mellon University, USA; moseleyb@andrew.cmu.edu*
[5] *Google, New York, USA; sergeiv@google.com*

ABSTRACT

The algorithms community has been modeling the underlying key features and constraints of massively parallel frameworks and using these models to discover new algorithmic techniques tailored to them. This monograph focuses on the Massively Parallel Model of Computation (MPC) framework, also known as the MapReduce model in the literature. It describes algorithmic tools that have been developed to leverage the unique features of the MPC framework. These tools were chosen for their broad applicability, as they can serve as building blocks to design new algorithms. The monograph is not exhaustive and includes topics such as partitioning and coresets, sample and prune, dynamic programming, round compression, and lower bounds.

Sungjin Im, Ravi Kumar, Silvio Lattanzi, Benjamin Moseley and Sergei Vassilvitskii (2023), "Massively Parallel Computation: Algorithms and Applications", Foundations and Trends® in Optimization: Vol. 5, No. 4, pp 340–417. DOI: 10.1561/2400000025.

1

Introduction

The modern era is witnessing a revolution in the ability to scale computations to massively large data sets. A key breakthrough in scalability was the introduction of fast and easy-to-use distributed programming models such as MapReduce (Dean and Ghemawat, 2008), Hadoop (hadoop.apache.org), and Spark (spark.apache.org). We refer to these programming models as **massively parallel frameworks**.

Massively parallel frameworks were originally designed for relatively simple types of computations such as counting the frequency of words in a data set. Since then, they have been shown to be useful for a far richer class of applications. The goal of a recent line of work is to study these frameworks algorithmically to unlock their true underlying power and expand their applicability. The hope is, through an algorithmic investigation, to achieve successes similar to those on topics such as cache-oblivious algorithms (Frigo *et al.*, 2012) and data streaming algorithms (McGregor, 2014).

Practically, massively distributed frameworks enable programmers to easily deploy algorithms on tens to thousands of machines. Algorithmically, the frameworks have restrictions on their computational expressive power to help ensure programs can be efficiently parallelized.

The challenges are then to (i) develop simple tools that reveal fundamentals of massive computation and aid algorithm design and (ii) understand which computations can benefit from the framework.

The algorithms community has been addressing this problem by modeling the underlying key features and constraints of massively parallel frameworks and using these models to discover new algorithmic techniques tailored to them. The first model of massively parallel computation was introduced for the MapReduce framework by Karloff *et al.* (2010) and several variants have been proposed since (Feldman *et al.*, 2010; Koutris *et al.*, 2018; Beame *et al.*, 2017; Andoni *et al.*, 2014; Goel and Munagala, 2012; Goodrich *et al.*, 2011; Pietracaprina *et al.*, 2012; Roughgarden *et al.*, 2016). Perhaps the main advantage of the model in Karloff *et al.* (2010) is its relative simplicity. It captures framework characteristics that are sufficient for algorithm design, without delving into the plethora of system parameters. In this monograph, we will primarily focus on this version of the model; we call it the Massively Parallel Model of Computation (MPC). See Section 2 for formal details.

The MPC model is a special case of the Bulk-Synchronous-Parallel (BSP) model of Valiant (1990), where machines have sublinear memory (i.e., n^δ for $\delta < 1$ and input size n) and computation proceeds in alternating **rounds** of communication and sequential computation. The MPC model can be thought of making different trade-offs than the classic PRAM computational model. Much of the difference comes from being able to run a sequential algorithm on a small sublinear portion of the data during a single round. Full details are given in Section 2.

The MPC model has a strong connection to practice and this is demonstrated by algorithmic developments resulting in good practical performance (Chierichetti *et al.*, 2010; Bahmani *et al.*, 2012a; Suri and Vassilvitskii, 2011; Karloff *et al.*, 2010; Mirzasoleiman *et al.*, 2013; Broder *et al.*, 2014; Feldman *et al.*, 2010; Zhao *et al.*, 2012; Ene *et al.*, 2011; Malkomes *et al.*, 2015; Kumar *et al.*, 2015; Bahmani *et al.*, 2012b; Ene and Nguyen, 2015; Cohen-Addad *et al.*, 2021b; Cohen-Addad *et al.*, 2021a; Lattanzi *et al.*, 2019; Ghaffari *et al.*, 2019b; Bateni *et al.*, 2017; Assadi *et al.*, 2019b; Bhaskara and Wijewardena, 2018) and influencing software libraries. For example, theoretical algorithms for k-means clustering have been incorporated in the Spark Machine Learning software library[1] (Bahmani *et al.*, 2012b).

1.1 Purpose of This Monograph

This line of work has demonstrated that massively parallel frameworks are useful for some challenging applications. With this as a proof-of-concept, an exciting area of research is to broaden the use of the frameworks to address a wide range of problems by using theoretical models to drive algorithm design.

This monograph will describe algorithmic tools that have been developed for massively distributed computing that leverage the unique features of the framework. The tools were chosen because we believe they are generally applicable and can be used as building blocks to design algorithms in the area.

This monograph is not exhaustive. However, it will cover the following areas.

- **Partitioning and Coresets:** This is one of the most natural approaches for parallel algorithms design. The idea is to partition the input to the problem across machines, and have each machine solve the problem on the individual parts. The individual solutions are then combined to build the solution to the overall problem.

- **Sample and Prune:** Another common approach to solve problems on large data sets is to use sampling to reduce problem size. Unfortunately, sampling from simple distributions, such as uniform, often misses too much information to solve a problem near optimally. We discuss the iterative sample-and-prune method, which has been shown to be efficient for many problems.

- **Dynamic Programming:** Dynamic programming is a powerful technique for solving problems. Unfortunately, it is typically difficult to parallelize. We discuss techniques for adapting certain dynamic programs to the massively parallel setting.

[1] https://spark.apache.org/docs/2.2.0/mllib-clustering.html

- **Rounds Reduction:** A simulation approach to solve problems in a parallel fashion is to apply a known algorithm, performing one step in a single round of distributed computation. While simple, it is often inefficient and leads to a large number of rounds. We discuss round compression, where multiple iterative rounds are compressed into a single round.

- **Lower Bounds:** Finally, we discuss the limitations of the massively parallel model of computation. We highlight the efforts to develop lower bounds for the model and derive connections to other models of computation.

1.2 Prerequisites

This monograph will assume the basics on approximation algorithm design and randomized algorithms. For a quick overview, we recommend the books by Williamson and Shmoys (2011, Chapter 2) and Mitzenmacher and Upfal (2005, Chapters 1-4).

References

Afrati, F. N., A. D. Sarma, S. Salihoglu, and J. D. Ullman. (2012). "Upper and Lower Bounds on the Cost of a Map-Reduce Computation". arXiv: 1206.4377.

Andoni, A., A. Nikolov, K. Onak, and G. Yaroslavtsev. (2014). "Parallel Algorithms for Geometric Graph Problems". In: *STOC*. 574–583.

Andoni, A., Z. Song, C. Stein, Z. Wang, and P. Zhong. (2018). "Parallel Graph Connectivity in Log Diameter Rounds". In: *FOCS*. 674–685.

Assadi, S., M. Bateni, A. Bernstein, V. S. Mirrokni, and C. Stein. (2019a). "Coresets Meet EDCS: Algorithms for Matching and Vertex Cover on Massive Graphs". In: *SODA*. 1616–1635.

Assadi, S., M. Bateni, and V. S. Mirrokni. (2019b). "Distributed Weighted Matching via Randomized Composable Coresets". In: *ICML*. 333–343.

Bahmani, B., R. Kumar, and S. Vassilvitskii. (2012a). "Densest Subgraph in Streaming and MapReduce". *PVLDB*. 5(5): 454–465.

Bahmani, B., B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. (2012b). "Scalable K-Means++". *PVLDB*. 5(7): 622–633.

Bateni, M. H., S. Behnezhad, M. Derakhshan, M. T. Hajiaghayi, R. Kiveris, S. Lattanzi, and V. Mirrokni. (2017). "Affinity clustering: Hierarchical clustering at scale". In: *NIPS*. 6867–6877.

Beame, P., P. Koutris, and D. Suciu. (2017). "Communication steps for parallel query processing". *JACM*. 64(6): 40:1–40:58.

Behnezhad, S., L. Dhulipala, H. Esfandiari, J. Lacki, and V. S. Mirrokni. (2019a). "Near-Optimal Massively Parallel Graph Connectivity". In: *FOCS*. 1615–1636.

Behnezhad, S., M. Hajiaghayi, and D. G. Harris. (2019b). "Exponentially Faster Massively Parallel Maximal Matching". In: *FOCS*. 1637–1649.

Bhaskara, A. and M. Wijewardena. (2018). "Distributed Clustering via LSH Based Data Partitioning". In: *ICML*. 569–578.

Bilardi, G., M. Scquizzato, and F. Silvestri. (2012). "A lower bound technique for communication on BSP with application to the FFT". In: *ECPP*. 676–687.

Broder, A. Z., L. G. Pueyo, V. Josifovski, S. Vassilvitskii, and S. Venkatesan. (2014). "Scalable K-Means by ranked retrieval". In: *WSDM*. 233–242.

Ceccarello, M., A. Pietracaprina, and G. Pucci. (2019). "Solving k-center Clustering (with Outliers) in MapReduce and Streaming, almost as Accurately as Sequentially". *Proc. VLDB Endow.* 12(7): 766–778.

Chang, Y., M. Fischer, M. Ghaffari, J. Uitto, and Y. Zheng. (2019). "The Complexity of $(\Delta{+}1)$ Coloring in Congested Clique, Massively Parallel Computation, and Centralized Local Computation". In: *PODC*. 471–480.

Charikar, M., W. Ma, and L.-Y. Tan. (2020). "New lower bounds for Massively Parallel Computation from query complexity". In: *SPAA*. 141–151.

Chierichetti, F., R. Kumar, and A. Tomkins. (2010). "Max-cover in map-reduce". In: *WWW*. 231–240.

Cohen-Addad, V., S. Lattanzi, S. Mitrovic, A. Norouzi-Fard, N. Parotsidis, and J. Tarnawski. (2021a). "Correlation Clustering in Constant Many Parallel Rounds". In: *ICML*. 2069–2078.

Cohen-Addad, V., S. Lattanzi, A. Norouzi-Fard, C. Sohler, and O. Svensson. (2021b). "Parallel and Efficient Hierarchical k-Median Clustering". In: *NeurIPS*.

Coy, S. and A. Czumaj. (2022). "Deterministic massively parallel connectivity". In: *STOC*. 162–175.

Czumaj, A., J. Lacki, A. Madry, S. Mitrovic, K. Onak, and P. Sankowski. (2018). "Round compression for parallel matching algorithms". In: *STOC*. 471–484.

Dean, J. and S. Ghemawat. (2008). "MapReduce: Simplified Data Processing on Large Clusters". *CACM*. 51: 107–113.

Ene, A., S. Im, and B. Moseley. (2011). "Fast clustering using MapReduce". In: *KDD*. 681–689.

Ene, A. and H. L. Nguyen. (2015). "Random Coordinate Descent Methods for Minimizing Decomposable Submodular Functions". In: *ICML*. 787–795.

Esfandiari, H., S. Lattanzi, and V. Mirrokni. (2018). "Parallel and Streaming Algorithms for K-Core Decomposition". In: *ICML*. 1396–1405.

Feldman, J., S. Muthukrishnan, A. Sidiropoulos, C. Stein, and Z. Svitkina. (2010). "On distributing symmetric streaming computations". *TALG*. 6(4): 66:1–66:19.

Frigo, M., C. E. Leiserson, H. Prokop, and S. Ramachandran. (2012). "Cache-Oblivious Algorithms". *TALG*. 8(1): 4.

Ghaffari, M., T. Gouleakis, C. Konrad, S. Mitrovic, and R. Rubinfeld. (2018). "Improved Massively Parallel Computation Algorithms for MIS, Matching, and Vertex Cover". In: *PODC*. 129–138.

Ghaffari, M., F. Kuhn, and J. Uitto. (2019a). "Conditional Hardness Results for Massively Parallel Computation from Distributed Lower Bounds". In: *FOCS*. *FOCS '19*.

Ghaffari, M., S. Lattanzi, and S. Mitrovic. (2019b). "Improved Parallel Algorithms for Density-Based Network Clustering". In: *ICML*. 2201–2210.

Ghaffari, M. and J. Uitto. (2019). "Sparsifying Distributed Algorithms with Ramifications in Massively Parallel Computation and Centralized Local Computation". In: *SODA*. 1636–1653.

Goel, A. and K. Munagala. (2012). "Complexity Measures for Map-Reduce, and Comparison to Parallel Computing". arXiv: 1211.6526.

Gonzalez, T. F. (1985). "Clustering to minimize the maximum intercluster distance". *TCS*. 38: 293–306.

Goodrich, M. T. (2010). "Simulating Parallel Algorithms in the MapReduce Framework with Applications to Parallel Computational Geometry". arXiv: 1004.4708.

Goodrich, M. T., N. Sitchinava, and Q. Zhang. (2011). "Sorting, Searching, and Simulation in the Mapreduce Framework". In: *ISAAC*. 374–383.

Hegeman, J. W. and S. V. Pemmaraju. (2015). "Lessons from the congested clique applied to MapReduce". *TCS*. 608: 268–281.

Im, S. and B. Moseley. (2015). "Brief Announcement: Fast and Better Distributed MapReduce Algorithms for k-Center Clustering". In: *SPAA*. 65–67.

Im, S. and B. Moseley. (2019). "A Conditional Lower Bound on Graph Connectivity in MapReduce". arXiv: 1904.08954.

Im, S., B. Moseley, and X. Sun. (2017). "Efficient massively parallel methods for dynamic programming". In: *STOC*. 798–811.

Indyk, P., S. Mahabadi, M. Mahdian, and V. S. Mirrokni. (2014). "Composable core-sets for diversity and coverage maximization". In: *PODS*. 100–108.

Jacob, R., T. Lieber, and N. Sitchinava. (2014). "On the complexity of list ranking in the parallel external memory model". In: *MFCS*. 384–395.

Karloff, H., S. Suri, and S. Vassilvitskii. (2010). "A Model of Computation for MapReduce". In: *SODA*. 938–948.

Koutris, P., S. Salihoglu, and D. Suciu. (2018). "Algorithmic Aspects of Parallel Data Processing". *Foundations and Trends in Databases*. 8(4): 239–370.

Kumar, R., B. Moseley, S. Vassilvitskii, and A. Vattani. (2015). "Fast Greedy Algorithms in MapReduce and Streaming". *TOPC*. 2(3): 14:1–14:22.

Lattanzi, S., T. Lavastida, K. Lu, and B. Moseley. (2019). "A framework for parallelizing hierarchical clustering methods". In: *ECML/PKDD*. 73–89.

Lattanzi, S., B. Moseley, S. Suri, and S. Vassilvitskii. (2011). "Filtering: a method for solving graph problems in MapReduce". In: *SPAA*. 85–94.

Lenzen, C. and R. Wattenhofer. (2010). "Brief announcement: exponential speed-up of local algorithms using non-local communication". In: *PODC*. 295–296.

Linial, N. (1987). "Distributive graph algorithms global solutions from local data". In: *FOCS*. 331–335.

Malkomes, G., M. J. Kusner, W. Chen, K. Q. Weinberger, and B. Moseley. (2015). "Fast Distributed k-Center Clustering with Outliers on Massive Data". In: *NIPS*. 1063–1071.

McGregor, A. (2014). "Graph stream algorithms: a survey". *SIGMOD Record*. 43(1): 9–20.

Mirzasoleiman, B., A. Karbasi, R. Sarkar, and A. Krause. (2013). "Distributed Submodular Maximization: Identifying Representative Elements in Massive Data". In: *NIPS*. 2049–2057.

Mitzenmacher, M. and E. Upfal. (2005). *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press.

Nemhauser, G. L., L. A. Wolsey, and M. L. Fisher. (1978). "An Analysis of Approximations for Maximizing Submodular Set Functions–I". *Math. Program.* 14(1): 265–294.

Park, H.-M., F. Silvestri, U. Kang, and R. Pagh. (2014). "MapReduce Triangle Enumeration With Guarantees". In: *CIKM*. 1739–1748.

Pietracaprina, A., G. Pucci, M. Riondato, F. Silvestri, and E. Upfal. (2012). "Space-round Tradeoffs for MapReduce Computations". In: *ICS*. 235–244.

Ponte Barbosa, R. da, A. Ene, H. L. Nguyen, and J. Ward. (2015). "The Power of Randomization: Distributed Submodular Maximization on Massive Datasets". In: *ICML*. Vol. 37. 1236–1244.

Ponte Barbosa, R. da, A. Ene, H. L. Nguyen, and J. Ward. (2016). "A New Framework for Distributed Submodular Maximization". In: *FOCS*. 645–654.

Roughgarden, T., S. Vassilvitskii, and J. T. Wang. (2016). "Shuffles and Circuits (On Lower Bounds on Massively Parallel Computation)". In: *SPAA '16*.

Suri, S. and S. Vassilvitskii. (2011). "Counting triangles and the curse of the last reducer". In: *WWW*. 607–614.

Valiant, L. G. (1990). "A bridging model for parallel computation". *CACM*. 33(8): 103–111.

Williamson, D. P. and D. B. Shmoys. (2011). *The Design of Approximation Algorithms*. Cambridge University Press.

Zhao, Z., G. Wang, A. Butt, M. Khan, V. Kumar, and M. Marathe. (2012). "SAHAD: Subgraph Analysis in Massive Networks Using Hadoop". In: *IPDPS*. 390–401.