LITERATURE REVIEW: — Dynamic Massive Parallel Computation Model for Graph Problems —

Isabelle Liu
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
isabelleliu@cmail.carleton.ca

November 2, 2019

1 Introduction

Big data and data analytics have become increasingly popular over the years, with the growing volume and variety of available data, and cheaper and more powerful computational processing. However, traditional data mining algorithms are becoming insufficient when performing computation on massive amounts of data, so parallel computing is introduced to increase the available computation power and to speed up application processing [1]. Parallel computing is a type of computing architecture where applications, computations, or processes are executed simultaneously. It basically breaks a larger task into several very similar smaller tasks so that they can be run independently at the same time, and their results are later combined upon completion. Parallel computing is widely used in various fields, such as climate prediction, fluid dynamics, and cryptology, in order to resolve their needs for speed.

One of the most popular models of parallel computing is the Massive Parallel Computation (MPC) model, which is seen as the canonical model for dealing with large scale data nowadays. It consists of machines with a large number of processors performing a set of coordinated tasks in synchronism [6]. And the main objective in the analysis of algorithms in the MPC model is to solve a problem with constant number of rounds while minimizing the amount of communication performed per round [5].

Although the MPC model is able to process massive amounts of data, it takes up large volumes of resources during computation. Also, because the datasets in the real world evolve continuously instead of being static, algorithms and models are forced to be re-executed after small modifications occur to the data, which leads to excessive processing time and resource requirements. As a result, researches of dynamic algorithms in traditional computing models are greatly promoted, since they are able to adjust and maintain solutions to given problems throughout input data modifications by performing only limited computation.

In this project, I aim to implement and test the dynamic algorithm in the Dynamic Massive Parallel Computation (DMPC) model by Italiano et al. [9] on an alternative graph-theoretic problem, the maximal independent set.

2 Literature Review

2.1 Classic MPC

The classic MPC model was first introduced in [10], and later refined. For a set of μ machines $M_1, ..., M_\mu$ that exchange messages in synchronous rounds, each machine is able to send and receive messages of up to the size of its local memory, which is S bits, at each round. We assume that $S, \mu \in O(N^{1-\epsilon})$, where N is the size of the input and ϵ is efficiently small, then the total communication per round is bounded by $S \cdot \mu \in O(N^{2-2\epsilon})$. For any MPC algorithms, there are three parameters that need to be bounded:

- Machine Memory: The total memory used by each machine per round is $O(N^{1-\epsilon})$.
- Total Memory: The total amount of data communicated per round is $O(N^{2-2\epsilon})$.
- Rounds: The number of rounds is $O(\log^i n)$, for a small $i \geq 0$.

2.2 MPC for Graph Problems

The concept of MPC is extensively studied in recents year. However, there exists a very important bottleneck in designing efficient MPC algorithms for graph problems when the space per machine is much smaller than the number of vertices n - it requires $\Omega(\log k)$ rounds to find a vertex at distance k from a given vertex [4]. For example, based on the 2-CYCLE conjecture, even distinguishing a graph consisting of single cycle of length n from 2 cycles of size $n/2$ requires $\Omega(\log n)$ rounds [11] [12]. Another one of the graph problem in the MPC model is the minimum spanning tree, which was recently proved to be solved in $O(\log n)$ rounds using an overall space of $\tilde{O}(n^{1+\epsilon})$ for some constant $\epsilon < 1$ [3].

It took almost 10 years for researchers to overcome the $O(\log n)$ barrier for computing approximate matching problems, and only very recently did Ghaffari and Uitto [8] implement an algorithm that can compute a $(1+\epsilon)$-approximate matching in $\tilde{O}(\sqrt{\log \Delta})$ rounds using sub-linear memory, where Δ is the maximum degree in the graph. Under the same memory assumption, Andoni et al. [2] showed that connected components was able to be run in $\tilde{O}(\log D)$ rounds, where D is the diameter of the graph. In addition, the computation of maximal independent set can be solved by the algorithm in [8] also in $\tilde{O}(\sqrt{\log \Delta})$ rounds. And this result is now improved by himself to simple $O(\log \log \Delta)$ round with $\tilde{O}(n)$ memory per machine [7].

2.3 Dynamic Algorithm for MPC

For a dynamic algorithm, the objective is to minimize the time spent and sometimes even the space required for updating the solution to a problem while the input gets modified. Unlike the classic MPC model, which has strictly sublinear memory of $\Omega(n)$ and requires $O(\log n)$ rounds to recompute a solution, the dynamic algorithm in the DMPC model hopes to establish some bounds for certain characteristics during recomputation:

- The number of machines active in each round
- The total amount of communication in each round
- The number of rounds required to update the solution

We can see that by bounding the number of active machines involved in the communication, we also imply the same bound on the amount of data that are sent in one round. While an ideal dynamic algorithm, which processes an input update in a constant number of rounds, using constant number of machines and constant amount of total communication, is often hard to achieve, a dynamic algorithm should at least use polynomially less resources than a static MPC model [9].

References

[1] Parallel computing — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Parallel_computing`, 2019. [Online; accessed 26-September-2019].

[2] A. Andoni, Z. Song, C. Stein, Z. Wang, and P. Zhong. Parallel graph connectivity in log diameter rounds. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 674–685, 2018.

[3] M. H. Bateni, S. Behnezhad, M. Derakhshan, M. T. Hajiaghayi, and V. Mirrokni. Massively parallel dynamic programming on trees, 2018.

[4] S. Behnezhad, L. Dhulipala, H. Esfandiari, J. Łącki, V. Mirrokni, and W. Schudy. Massively parallel computation via remote memory access. In *The 31st ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '19, pages 59–68, New York, NY, USA, 2019. ACM.

[5] D. Durfee, L. Dhulipala, J. Kulkarni, R. Peng, S. Sawlani, and X. Sun. Parallel batch-dynamic graphs: Algorithms and lower bounds, 2019.

[6] S. E. Fahlman, G. E. Hinton, and T. J. Sejnowski. Massively parallel architectures for AI: Netl, thistle, and boltzmann machines. pages 109–113, 1983.

[7] M. Ghaffari, T. Gouleakis, C. Konrad, S. Mitrović, and R. Rubinfeld. Improved massively parallel computation algorithms for MIS, matching, and vertex cover, 2018.

[8] M. Ghaffari and J. Uitto. Sparsifying distributed algorithms with ramifications in massively parallel computation and centralized local computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 1636–1653, Philadelphia, PA, USA, 2019. Society for Industrial and Applied Mathematics.

[9] G. F. Italiano, S. Lattanzi, V. S. Mirrokni, and N. Parotsidis. Dynamic algorithms for the massively parallel computation model. In *The 31st ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '19, pages 49–58, New York, NY, USA, 2019. ACM.

[10] Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for mapreduce. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 938–948, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

[11] T. Roughgarden, S. Vassilvitskii, and J. R. Wang. Shuffles and circuits: (on lower bounds for modern parallel computation). In *Proceedings of the 28th ACM Symposium*

on Parallelism in Algorithms and Architectures, SPAA '16, pages 1–12, New York, NY, USA, 2016. ACM.

[12] G. Yaroslavtsev and A. Vadapalli. Massively parallel algorithms and hardness for single-linkage clustering under ℓ_p-distances. In *35th International Conference on Machine Learning*, ICML '18, 2018.