Front. Comput. Sci., 2024, 18(4): 184611
https://doi.org/10.1007/s11704-024-3452-2

LETTER

Massively parallel algorithms for fully dynamic all-pairs shortest paths

Chilei WANG[1], Qiang-Sheng HUA (✉)[1], Hai JIN[1], Chaodong ZHENG[2]

1 National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

2 State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China

1 Introduction

In recent years, the Massively Parallel Computation (MPC) model has gained significant attention. However, most of distributed and parallel graph algorithms in the MPC model are designed for static graphs [1]. In fact, the graphs in the real world are constantly changing. The size of the real-time changes in these graphs is smaller and more localized. *Dynamic graph algorithms* [2,3] can deal with graph changes more efficiently [4] than the corresponding static graph algorithms. However, most studies on dynamic graph algorithms are limited to the single machine model. Moreover, a few parallel dynamic graph algorithms (such as the graph connectivity) in the MPC model [5] have been proposed and shown superiority over their parallel static counterparts.

To the best of our knowledge, there are no existing dynamic all-pairs shortest paths (APSP) algorithms working in the MPC model. We aim to design a parallel dynamic algorithm that is faster than all the existing static parallel APSP algorithms. Our contributions can be summarized as follows:

1) We propose the first parallel fully dynamic APSP algorithm in the MPC model.
2) Our parallel algorithm has a lower round complexity than the existing fastest MPC algorithm.

2 Problem definition

In this paper, we consider directed weighted graphs $G=(V,E,W)$ without negative cycles, where V denotes the set of nodes, E is the set of edges, and $W: E \to \mathbb{R}$ is a weighted function. $|V|=n$ is the number of nodes in the graph G. The problem studied in this paper is defined below (The definition of the MPC model and other related definitions are given in Online Resource):

Definition 1 In the MPC model with the memory of each processor as $\tilde{O}(n^{\alpha})$, where $\alpha \in (0,1)$ is a constant. Given a graph above, the update operations are deleting or inserting nodes and their incident edges. The parallel fully dynamic APSP problem is to maintain the shortest distances and paths between nodes of a graph in the MPC model.

Received June 1, 2023; accepted January 10, 2024

E-mail: qshua@hust.edu.cn

3 Fully dynamic APSP algorithm in the MPC model

At a high-level, our algorithm can be seen as a parallel variant of the sequential dynamic APSP algorithm proposed by Abraham et al. [2]. Due to the space limit, we present only the challenges encountered and techniques used in parallelizing Abraham et al.'s algorithm. The complete design and detailed description of our parallel algorithm are provided in Online Resource.

3.1 The sequential algorithm of [2]

The sequential algorithm of [2] consists of three components: the preprocessing procedure, the decremental procedure and the incremental procedure.

The preprocessing procedure is used to process the current graph and obtain an efficient data structure. In [2], the shortest paths between nodes were divided into $\lceil \log n \rceil$ hierarchies since the number of edges on a path can be at most n.

In the ith hierarchy ($i \in \{1,2,\ldots,\lceil \log n \rceil\}$), a congestion value of zero is initially assigned to each node in the graph. The congestion value of a node is the number of shortest paths with at most 2^i edges that contain the node in the ith hierarchy. Then, a sampling strategy [2] is used to select a node subset with size $O(\frac{n \ln n}{2^i})$, forming an initial hitting set. Next, they computed the shortest paths with at most 2^i edges from and to a node u in the sampled set with the Bellman-Ford algorithm according to the congestion value size of nodes.

Utilizing these shortest paths, the congestion value of each node and the distances between at most n^2 pairs of nodes that pass through u were computed. Nodes with large congestion values, not originally in the sampled set, were added to the sampled set as the final hitting set at the end of the ith hierarchy, after computing shortest paths for all sampled

nodes. After iterating over all hierarchies, $\lceil \log n \rceil$ hitting sets and the corresponding distances and paths between nodes that pass through vertices of the hitting sets were obtained. Storing these distances may require up to $O(n^3 \log n)$ memory.

The decremental procedure in [2] only needs to deal with affected nodes whose shortest paths are destroyed when nodes are deleted. This can be done by using the shortest paths with roots in the hitting set obtained in the preprocessing procedure above. For each affected node, a new graph with related edges is constructed, and Dijkstra algorithm is used to compute the shortest paths from and to this node. Finally, the new distances between nodes are compared with the remaining distances to obtain the shortest distances between any two nodes.

For the insertion of nodes, a modified Floyd-Warshall algorithm is used to compute the distance matrix. Ultimately, using a standard strategy proposed in [4], Abraham et al. [2] obtained a fully dynamic APSP algorithm with a worst-case update time of $O(n^{2+2/3} \log^{4/3} n)$.

3.2 The main challenges

However, employing Abraham et al.'s algorithm [2] directly in the MPC model would result in a round complexity of $\tilde{O}(n^{2+2/3})$, which is too high and unrealistic. The main challenges are as follows:

1. The preprocessing procedure of [2] requires $O(n^3 \log n)$ memory to store the distances between all vertices, which is too large. Reducing the memory and round complexities while still storing these distances is a significant challenge.
2. For the decremental procedure in [2], constructing a new graph for every affected node in each hierarchy and still using Dijkstra's algorithm would result in a high round complexity.
3. In [2], there is a tight connection of the computation and comparison of shortest distances between the preprocessing procedure and the decremental procedure. If we reduce the total memory needed for the preprocessing procedure, it may pose new challenges to design the decremental procedure.
4. The incremental procedure in [2] uses a modified Floyd-Warshall algorithm with two-layer loops to update the node insertions one by one. However, implementing this algorithm directly in the MPC model would result in an unacceptable increase in round complexity.

3.3 The techniques used in our parallel algorithm

3.3.1 The parallel preprocessing procedure

To address the first challenge mentioned in Section 3.2, we remove the computation of distances between nodes during our parallel preprocessing procedure and only store the shortest path trees. This reduces the round complexity required to compute the distances between vertices, and the memory required to store the trees is only $\tilde{O}(n^2)$ since these trees can be used to calculate these distances (refer to Algorithm 2 in Online Resource).

3.3.2 The parallel decremental procedure

In tackling the second challenge in Section 3.2, we only concentrate on the nodes in the hitting set associated with the affected nodes in the parallel decremental procedure. We replace the Dijkstra algorithm with the restricted Bellman-Ford algorithm proposed in [1]. Due to resource contention, the restricted Bellman-Ford algorithm (refer to Algorithm 1 in Online Resource) can only compute the shortest path tree of a single node at a time, potentially resulting in a significant round complexity. Subsequently, we introduce an algebraic method to reduce the round complexity and update the distances and shortest paths between nodes accurately.

This algebraic method combines the blocked Floyd-Warshall algorithm [6] and the sampling strategy of [2], using the short-hop distances obtained by the restricted Bellman-Ford algorithm to compute the long-hop distances between nodes. The short-hop (or long-hop) distance means that the number of edges on the shortest path of any two nodes is small (or large).

The third challenge in Section 3.2 arises from removing the calculation of distances between nodes in the parallel preprocessing procedure in Section 3.3.1, which increases the round complexity when performing this step in the decremental procedure. To address this problem, we utilize matrix multiplication on a semiring [7] to calculate the distances between nodes within each hierarchy and compare them between these hierarchies (refer to Algorithm 3 in Online Resource).

3.3.3 The parallel incremental procedure

To address the fourth challenge in Section 3.2, we combine the blocked Floyd-Warshall algorithm [6] with an MPC algorithm of matrix multiplication on a semiring [7] in our parallel incremental procedure. This approach reduces the round complexity when computing both the shortest distances and paths (refer to Algorithm 4 in Online Resource).

4 The main results

In this section, we will present the round complexity of our parallel dynamic APSP algorithm (refer to Theorem 1) and the performance comparison between our parallel algorithm and other existing works (refer to Table 1). The proof of Theorem 1 and the detailed analysis of Table 1 are provided in the Online Resource.

Theorem 1 For the problem in Definition 1, there exists a parallel fully dynamic randomized APSP algorithm with $O(n^{\frac{2}{3}-\frac{\alpha}{6}} \log n/\alpha)$ worst-case update rounds and $O(n^{3-\alpha/2})$ total memory, with high probability.

5 Conclusion

In this paper, we propose the first fully dynamic parallel all-pairs shortest path algorithm in the MPC model with a worst-case update rounds of $O(n^{\frac{2}{3}-\frac{\alpha}{6}} \log n/\alpha)$. We compare our algorithm with the existing static APSP algorithms in the MPC model, demonstrating the efficiency of our approach.

Table 1 Comparing our parallel fully dynamic APSP algorithm with the existing works

	Rounds	Memory	CC	EW	Query	Type
Karczmarz et al. [8]	$O(d), (d \in [1,n])$	$\tilde{O}(n^3)$	$\tilde{O}(n^3)$	Real	Distances	Deterministic
Cao et al. [9]	$O(n^{3/2+o(1)})$	$\tilde{O}(n^3)$	$\tilde{O}(n^3)$	Integer	Distances/Paths	Randomized
AEV of Hajiaghayi et al. [7]	$O(n/\alpha)$	$O(n^{3-\alpha/2})$	$O(n^3 \log n)$	Real	Distances/Paths	Deterministic
ADP of Abraham et al. [2]	$\tilde{O}(n^{2+2/3})$	$O(n^{\alpha})^*$	$\tilde{O}(n^{2+2/3})$	Real	Distances/Paths	Randomized
This work	$\boldsymbol{O(n^{\frac{2}{3}-\frac{\alpha}{6}} \log n/\alpha)}$	$\boldsymbol{O(n^{3-\alpha/2})}$	$\boldsymbol{O(n^3 \log n)}$	**Real**	**Distances/Paths**	**Randomized**

Remark: (1) AEV: an extended version, ADP: a direct parallelization, CC: computation complexity, EW: edge weight; (2) * means that the total memory required is the sum of memory of $O(1)$ machines.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 61972447 and 61832006).

Competing interests The authors declare that they have no competing interests or financial conflicts to disclose.

Supporting information The supporting information is available online at journal.hep.com.cn and link.springer.com.

References

1. Dinitz M, Nazari Y. Massively parallel approximate distance sketches. In: Proceedings of the 23rd International Conference on Principles of Distributed Systems. 2019, 35: 1−35: 17
2. Abraham I, Chechik S, Krinninger S. Fully dynamic all-pairs shortest paths with worst-case update-time revisited. In: Proceedings of the 28th Annual ACM SIAM Symposium on Discrete Algorithms. 2017, 440−452
3. Zhu X, Li W, Yang Y, Wang J. Incremental algorithms for the maximum internal spanning tree problem. Science China Information Sciences, 2021, 64(5): 152103
4. Henzinger M R, King V. Maintaining minimum spanning forests in dynamic graphs. SIAM Journal on Computing, 2001, 31(2): 364–374
5. Italiano G F, Lattanzi S, Mirrokni V S, Parotsidis N. Dynamic algorithms for the massively parallel computation model. In: Proceedings of the 31st ACM Symposium on Parallelism in Algorithms and Architectures. 2019, 49−58
6. Sao P, Kannan R, Gera P, Vuduc R W. A supernodal all-pairs shortest path algorithm. In: Proceedings of the 25th SIGPLAN Symposium on Principles and Practice of Parallel Programming. 2020, 250−261
7. Hajiaghayi M, Lattanzi S, Seddighin S, Stein C. MapReduce meets fine-grained complexity: MapReduce algorithms for APSP, matrix multiplication, 3-SUM, and beyond. 2019, arXiv preprint arXiv: 1905.01748
8. Karczmarz A, Sankowski P. A deterministic parallel APSP algorithm and its applications. In: Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms. 2021, 255−272
9. Cao N, Fineman J T. Parallel exact shortest paths in almost linear work and square root depth. In: Proceedings of 2023 ACM-SIAM Symposium on Discrete Algorithms. 2023, 4354−4372