A Parallel Framework for Efficiently Updating Graph Properties in Large Dynamic Networks

Arindam Khanda
akkcm@mst.edu
Missouri University of Science and Technology
Rolla, MO, USA

Sajal K. Das
sdas@mst.edu
Missouri University of Science and Technology
Rolla, MO, USA

ABSTRACT

Graph queries on large networks leverage the stored graph properties to provide faster results. Since real-world graphs are mostly dynamic, i.e., the graph topology changes over time, the corresponding graph attributes also change over time. In certain situations, recompiling or updating earlier properties is necessary to maintain the accuracy of a response to a graph query. Here, we first propose a generic framework for developing parallel algorithms to update graph properties on large dynamic networks. We use our framework to develop algorithms for updating Single Source Shortest Path (SSSP) and Vertex Color. Then we propose applications of the developed algorithms in Unmanned Aerial Vehicle (UAV) based delivery systems under time-varying dynamics. Finally, we implement our SSSP and vertex color update algorithms for Nvidia GPU architecture and show empirically that the developed algorithms can update properties in large dynamic networks faster than the state-of-the-art techniques.

CCS CONCEPTS

• **Computing methodologies** → **Parallel algorithms**.

KEYWORDS

datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
Arindam Khanda and Sajal K. Das. 2023. A Parallel Framework for Efficiently Updating Graph Properties in Large Dynamic Networks. In *ICDCN 2023: 24th International Conference on Distributed Computing and Networking, Jan 04–07, 2023, IIT Kharagpur, India.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3571306.3571359

1 INTRODUCTION

Complex system analysis often takes the help of network modeling, where the interacting entities are modeled as vertices and the interactions are mapped as the edges. Properties of the modeled network help to find interesting insights about the actual system. As a result, network analysis has applications in various domains including bioinformatics, drug discovery, internet routing, and recommendation systems. For large networks, computing and storing graph properties efficiently is itself an expensive operation due to the irregular memory access during graph traversal. The real-world networks, which are dynamic in nature, possess additional challenges of maintaining the correct property values with the change in network structure. Most of the existing algorithms were developed targeting static graphs and thus if applied on dynamic networks, they recompute properties on network snapshots at different time instances. This technique of applying static graph algorithms on dynamic networks is computationally expensive and involves redundant operations. Furthermore, the challenge increases with the increase in the size of the network[1–3].

Here, we propose a framework to develop algorithms to update graph properties efficiently.

Problem Statement: Let $G_t(V_t, E_t)$ be the graph at time step t and $u.prop_t$ be the corresponding graph property (E.g., distance from source in SSSP or color assignment in vertex coloring problem) of a vertex u. Let $\Delta E_t = E_{t+1} - E_t$ be the set of changed edges from time step t to time step $t+1$. It consists of two subsets, Ins_t and Del_t, respectively the set of inserted edges and deleted edges at time step t. Thus, $E_{t+1} = ((E_t \cup Ins_t) \setminus Del_t)$. Our *goal* is to efficiently compute the updated property $u.prop_{t+1}$ for all $u \in V_{t+1}$, without recomputing from scratch.

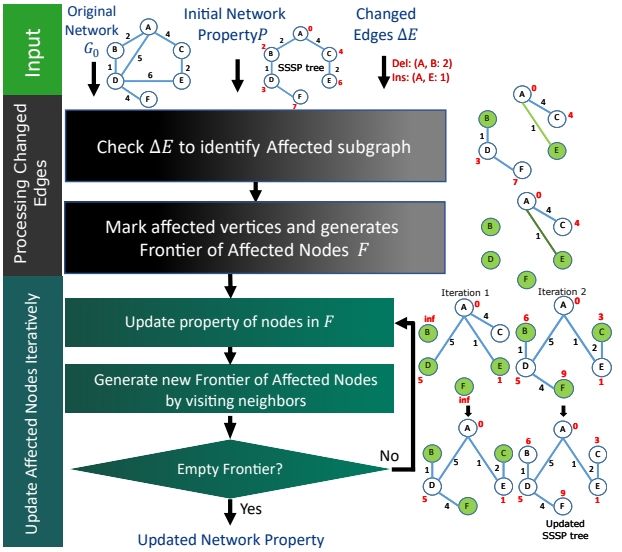


Figure 1: Graph property update framework demonstrated with an example of SSSP update.

2 GRAPH PROPERTY UPDATE FRAMEWORK

This section provides details on the proposed approach.

Step 1 identifies the affected end vertices of each edge $(u, v) \in \Delta E$. In this edge-centric parallel operation, the affected vertices directly related to the changed edges are gathered in a frontier for further

ICDCN 2023, Jan 04–07, 2023, IIT Kharagpur, India

ACM ISBN 978-1-4503-9796-4/23/01.
https://doi.org/10.1145/3571306.3571359

processing. **Step 2** is an iterative process of updating the property. In each iteration, the property is updated in parallel for the affected vertices in the current frontier. The next frontier is generated by visiting the neighbors and selecting the possible set of affected vertices. The iterative process converges and achieves correctness when the frontier becomes empty.

Fig. 1 describes our parallel framework with an example of SSSP update. In the example, the green circles are the affected vertices at different stages of SSSP update algorithm. For more details on SSSP update see [3]. Similar to SSSP update, our framework can be used to develop a vertex color update algorithm, where the initial color can be corrected by finding the affected vertices and recoloring them. In an asynchronous parallel framework, recoloring may produce color conflict and it can be solved iteratively in step 2.

3 APPLICATIONS OF OUR FRAMEWORK

3.1 Centralized Drone-based Delivery

In a centralized drone-based delivery system, drones follow the delivery route provided by the central server. In varying wind conditions (wind speed and direction change with time), finding the most efficient delivery routes becomes a problem of finding SSSP in a dynamic network. In [2], we applied our framework to efficiently update the delivery route of the drones in a centralized system under time-varying dynamics.

3.2 Drone Truck Co-operated Delivery

In [1], we proposed a drone truck co-operated delivery scenario, where drones start from a delivery truck, perform their delivery, and return to the truck. In this setup, we consider a fixed predefined route and rest areas for the truck. However, each drone's delivery route is dynamic due to time-dependent factors such as wind direction. We use the SSSP update algorithm to solve a multi-objective problem where the delivery system aims to complete a set of deliveries in the minimum time while the drones try to minimize their energy consumption to meet the limited battery constraint.

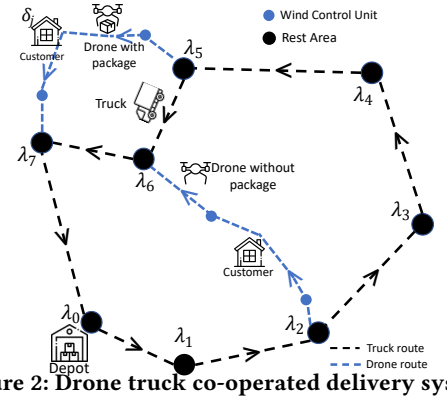


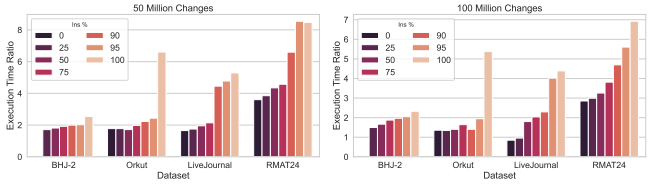
Figure 2: Drone truck co-operated delivery system.

4 EXPERIMENTAL RESULTS

We use our framework to develop and implement parallel algorithms to update SSSP and Vertex Color in large dynamic networks. In our experiment, $x\%$ *Ins* in ΔE indicates total $\Delta E * x/100$ edge insertion and $\Delta E * (1 - x/100)$ edge deletion.

4.1 Single Source Shortest Path Update

Our NVIDIA CUDA-based SSSP update implementation[3] outperforms state-of-the-art Gunrock's [5] GPU-based SSSP implementation (recomputation approach) in most cases. Fig. 3a, 3b shows the ratio of SSSP recomputation [5] time, and SSSP update time on GPU.

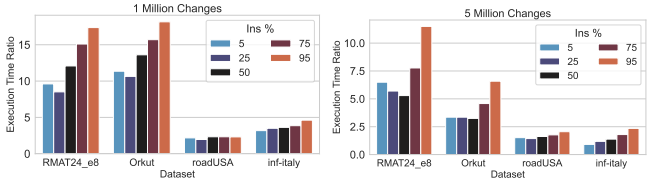


(a) 50 million change edges **(b) 100 million change edges**

Figure 3: SSSP: Comparison with Gunrock's implementation.

4.2 Vertex Color Update

Fig. 4a, and 4b show the ratio of color recomputation [4] time (using GPU-based Kokkos coloring), and vertex color update time on GPU. Experimental result shows that the execution ratio is more than 1 in most cases, i.e., the update algorithm takes less time than recomputation.



(a) 1 million change edges **(b) 5 million change edges**

Figure 4: Color Update: Comparison with static coloring[4]

5 CONCLUSION

We introduce a parallel framework to develop algorithms for updating the properties of large dynamic networks. Using our framework, we develop and implement SSSP and vertex color update algorithms on GPU architecture. Empirical results show that our implementations update different graph properties faster than the current state-of-the-art methods. We have already applied SSSP update algorithm in UAV-based delivery systems for efficient delivery route selection under time-varying dynamics. We plan to apply our vertex color update algorithm in a dynamic delivery-scheduling scenario.

ACKNOWLEDGMENTS

This work was supported by the NSF projects SANDY (Award # OAC-1725755) and CANDY (Award # OAC-2104078)

REFERENCES

[1] Arindam Khanda, Federico Corò, and Sajal K Das. 2022. Drone-Truck Cooperated Delivery Under Time Varying Dynamics. In *Proceedings of the 2022 Workshop on Advanced tools, programming languages, and PLatforms for Implementing and Evaluating algorithms for Distributed systems*. 24–29.

[2] Arindam Khanda, Federico Corò, Francesco Betti Sorbelli, Cristina M. Pinotti, and Sajal K. Das. 2021. Efficient Route Selection for Drone-based Delivery Under Time-varying Dynamics. In *IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*. 437–445. https://doi.org/10.1109/MASS52906.2021.00061

[3] Arindam Khanda, Sriram Srinivasan, Sanjukta Bhowmick, Boyana Norris, and Sajal K. Das. 2022. A Parallel Algorithm Template for Updating Single-Source Shortest Paths in Large-Scale Dynamic Networks. *IEEE Transactions on Parallel and Distributed Systems* 33, 4 (2022), 929–940. https://doi.org/10.1109/TPDS.2021.3084096

[4] Sivasankaran Rajamanickam, Seher Acer, Luc Berger-Vergiat, Vinh Dang, Nathan Ellingwood, Evan Harvey, Brian Kelley, Christian R Trott, Jeremiah Wilke, and Ichitaro Yamazaki. 2021. Kokkos Kernels: Performance Portable Sparse/Dense Linear Algebra and Graph Kernels. *arXiv preprint arXiv:2103.11991* (2021).

[5] Yangzihao Wang, Andrew Davidson, Yuechao Pan, Yuduo Wu, Andy Riffel, and John D Owens. 2016. Gunrock: A high-performance graph processing library on the GPU. In *Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. 1–12.