[1]Muteb Alshammari

Innovations in Dynamic Minimum Spanning Tree Algorithms: A Comprehensive Review

Journal of Electrical Systems

Abstract: - The paper provides a comprehensive overview of recent advancements in the field of maintaining Minimum Spanning Tree (MST) in fully dynamic graphs. It explores various aspects of dynamic graph processing, including memory-bound nature, representation learning, data structures optimized for GPU, visualization challenges, and the application of dynamic graphs in software testing frameworks and graph neural networks. The paper highlights the collective contributions of the discussed papers in pushing the boundaries of efficiency, applicability, and performance in the dynamic maintenance of MST across various computational models and practical applications.

Keywords: Minimum Spanning Tree, MST, Dynamic Algorithms, Combinatorics, Graph Theory.

I. INTRODUCTION

Recent advancements in dynamic algorithms for maintaining minimum spanning tree (MST) have been significant, addressing both deterministic and randomized approaches, as well as applications in various computational models and settings. Dynamic graphs are characterized by their ability to model and process relationships that change over time, capturing the evolving nature of various real-world phenomena. This capability is crucial for applications ranging from social network analysis to traffic forecasting and brain functionality modeling, as it allows for the efficient handling of systems where both the structure and the interactions within the system evolve over time [2], [15], [3].

One of the fundamental characteristics of dynamic graphs is their capacity to integrate both topological and temporal information, offering a compact representation of dynamic systems. This integration is crucial for applications ranging from social network analysis to traffic forecasting and brain functionality modeling, as it allows for the efficient handling of systems where both the structure and the interactions within the system evolve over time [2], [15], [3]. Another important characteristic of dynamic graphs is their dependency on memory bandwidth, which underscores the performance constraints imposed by the rate of memory access on the underlying hardware. This issue becomes especially problematic as dynamic graph algorithms are used on progressively larger datasets, putting significant pressure on memory systems and leading to reduced performance. The need for research into leveraging new memory technologies to enhance the efficiency of dynamic graph-based processing is underscored by these challenges [5]. Dynamic graph representation learning has also emerged as a significant area of interest, focusing on capturing the evolutionary essences of graphs, such as temporal and repetitive information. This involves novel methodologies that address the neglect of repetitive interactions between nodes, which are crucial for accurate node representation and for understanding the evolving patterns of these interactions [4].

Furthermore, advancements in dynamic graph data structures, such as those optimized for GPU, have significantly improved insertion and deletion rates, supporting faster and more efficient updates and queries [5]. The visualization of dynamic graphs presents its own set of challenges, necessitating scalable approaches that can adapt to the graphs' time varying properties and provide insights into their structure and evolution.

In addition, the development of concurrent dynamic graph algorithms highlights the importance of exploiting concurrency for applications that require dynamic updates, demonstrating the scalability and limitations of static graph analytics methods in dynamic settings [1]. Lastly, the application of dynamic graphs in software testing frameworks and the modeling of dynamic information in graph neural networks illustrate the broad applicability and potential of dynamic graphs in enhancing the performance of graph analytical tasks and reflecting changing trends in data over time [14], [17].

[1] *Corresponding author: Department of Information Technology, Faculty of Computing and Information Technology, Northern Border University, Arar, Saudi Arabia. Email: Muteb.Alshammari@nbu.edu.sa

This paper is structured as follows: Section 2 reviews deterministic fully dynamic graph algorithms, highlighting key contributions and their performance metrics. Section 3 discusses randomized approaches and their applications in dynamic graph settings. Section 4 concludes the paper and synthesizes the insights gained from the review, emphasizing the critical role of dynamic graph processing in real-world applications and the need for ongoing research.

II. Deterministic Fully Dynamic Graph Algorithms

A. Minimum Spanning Forest Maintenance

In [8], Eppstein et al. introduced an efficient algorithm designed to maintain a minimum spanning forest (MSF) of a plane graph while accommodating online modifications. The algorithm in the paper utilizes dynamic tree data structures (edge-ordered dynamic tree) to efficiently maintain an MSF of a plane graph. The edge-ordered dynamic tree data structure facilitates a variety of operations, such as finding the minimum or maximum edge, the root, or the lowest common ancestor of two nodes, which are crucial for managing the graph's spanning trees. It utilizes the subdivision representation scheme of Guibas and Stolfi [9] for describing and manipulating the dynamic plane graph, providing a foundation for complex operations such as edge insertions or deletion. Furthermore, balanced trees serve as supplementary data structures for preserving node routes during splits and merges, hence facilitating the efficient resolution of queries in logarithmic time. The approach described in the paper can efficiently update and maintain a minimum spanning forest of a plane network with n vertices in O(log n) time per update. While the paper discusses maintaining an MST in a dynamic setting, it acknowledges that the modifications permitted are limited by the embedding, suggesting that the approach may not be flexible for all types of graph modifications such as insertion of new vertices.

Holm et al. in [12] introduced deterministic algorithms or efficiently managing graph connectivity, MSF, 2-edge connectivity, and bi-connectivity. These algorithms start with an empty graph and achieve amortized operation costs that scale with the number of vertices. They introduced a novel approach to updating dynamic graphs, focusing on maintaining accurate connectivity information through operations like insertions and deletions of edges, with a detailed explanation of the amortized operation costs. The research expands upon previously established methods, enhancing the most well-known deterministic and randomized bounds for dynamic connectivity and minimum spanning tree problems by the implementation of algorithms that are both more straightforward and more effective.

It achieves a breakthrough in operation costs, presenting an algorithm that updates the graph's connectivity information at a cost that scales poly-logarithmically with the number of vertices, marking a substantial improvement over the best known deterministic and randomized bounds for these problems. The proposed approach takes $O(\log^2 n)$ amortized cost for connectivity and $O(\log^4 n)$ amortized cost for 2-edge connectivity, bi-connectivity, and minimum spanning forest.

Holm et al. in [13] introduced a novel data structure for solving the fully-dynamic MSF in dynamic graphs. Their approach achieves an amortized update time of $O(\log^4 n \log \log n)$. This paper examines the correlation between the data structures used for decremental MSF and fully dynamic connectivity. It also addresses the difficulties in converting the fully dynamic connectivity structure with a time complexity of $O(\log^2 n \log \log n)$ into an enhanced decremental MSF structure with the same time complexity.

The paper additionally presents a comprehensive performance analysis of the data structure, elucidating the time complexity associated with several operations including initialization, edge level increases, and edge removals. In summary, the study provides a thorough investigation of the fully dynamic minimal spanning forest problem and the creation of a high-performing data structure to tackle it.

In [22], Tom and his colleagues conducted research on dynamic algorithms for hierarchical agglomerative clustering (HAC) on networks with weighted edges. The authors present a parallel batch-dynamic technique for MSF, which is capable of responding to questions regarding clusters formed by single-linkage graph HAC. Additionally, they investigate the effectiveness of dynamic graph HAC when edge insertions and deletions occur. The study introduces a parallel batch-dynamic MSF technique that achieves a low estimated amortized cost. It emphasizes the importance of this approach beyond just clustering. In addition, the authors present lower constraints for dynamic and partially dynamic graph HAC using complete linkage, weighted average linkage, and average linkage. These lower bounds are derived from conjectures such as the strong exponential temporal hypothesis

(SETH). The research also examines parallel primitives employed in their methods and introduces a data format for relative-error quantile summary to address parallel batch-decremental MSF.

The paper commences by presenting HAC techniques, which arrange data into meaningful groups. The significance of investigating dynamic HAC on edge-weighted graphs is underscored by the presence of vast and swiftly evolving contemporary datasets. The authors suggest a parallel batch dynamic approach for managing minimal spanning forests (MSFs) and showcase its effectiveness in handling batches of edge updates with low predicted amortized work and span. Furthermore, the work explores the difficulties associated with dynamic graph HAC and gives lower limits for different linkage functions based on conjectures such as the SETH, offering valuable insights into the complexity of dynamic HAC algorithms. The approach provides theoretical assurances of $O(k \log^6 n)$ expected amortized work and $O(k \log^4 n)$ span with a high likelihood on a batch of k edge insertions or removals.

Table 1:Summary of Major Research Contributions in The Deterministic Dynamic MST.

Paper Title and Authors	**Main Contribution**	**Cost**	**Year**
Maintenance of A Minimum Spanning Forest In A Dynamic Plane Graph by David, Giuseppe, Tamassia, Tarjan, Ery, and Moti	Efficient algorithms for maintaining minimum spanning forest in dynamic planar graph. Also, development of edge-ordered dynamic tree data structure for algorithm implementation.	$O(\log n)$	1992
Maintaining Minimum Spanning Trees in Dynamic Graphs by Henzinger and King.	First deterministic and fully dynamic algorithm for maintaining MST with an amortized time complexity. It applies the sparsification technique to reduce the running time for sequences of updates, further improving the efficiency of the algorithm.	$O(n^{1/3} \log n)$	1997
Competitive Maintenance of Minimum Spanning Trees in Dynamic Graphs by Dynia, Korzeniowski, and Kutyłowski	Introduced the concept of competitive maintenance of MST in dynamic graphs, emphasizing the significance of minimizing changes in the MST between rounds and its application in mobile networks and scenarios where changes in the minimum spanning tree are costly.	$O(n \log n)$	2007
Poly-Logarithmic Deterministic Fully Dynamic Algorithms for Connectivity, Minimum Spanning Tree, 2-Edge, and Biconnectivity by Holm, Lichtenberg, and Thorup	Fully dynamic graph problems addressed: connectivity, minimum spanning forest, 2-edge connectivity, biconnectivity.	$O(\log^4 n)$	2001
Experimental Analysis of Algorithms for Updating Minimum Spanning Trees on Graphs Subject to Changes on Edge Weights by Ribeiro and Toso	DR-trees and DRD-trees data structures to maintain dynamic MST. Also, comprehensive performance evaluation of tree-based data structures and algorithms in the context of graph processing, with a particular focus on scalability and efficiency as the size of the graph increases.	$O(\|E\|)$	2007
Efficient Maintenance of Minimum Spanning Trees in Dynamic Weighted Undirected Graphs by Mao, Huigang, Xinyun, Xiong, Xia, and Ke	Presented an algorithm for efficiently maintaining the minimum spanning tree (MST) in dynamic graphs, providing a detailed analysis of the algorithm's time complexity and correctness.	$O(\log^4 n)$	2024
Dynamic Graph Neural Networks by Yao, Ziyi, Ren, Zhao, Tang, and Yin	Introduced dynamic graph neural networks.	Not specified	2018
Faster Fully-Dynamic Minimum Spanning Forest by Holm, Rotenberg, and Wulff-Nilsen	New data structure for fully-dynamic MSF, achieving improved amortized time and providing a detailed analysis of the data structure.	$O(\log^4 n \log \log n)$	2015

Parallel Batch-Dynamic Minimum Spanning Forest and The Efficiency of Dynamic Agglomerative Graph Clustering by Tseng, Dhulipala, and Shun	Introduced a parallel batch-dynamic algorithm for maintaining minimum spanning forests (MSFs) and explored the efficiency of dynamic graph hierarchical agglomerative clustering (HAC) under edge insertions and deletions, achieving low expected amortized cost and providing lower bounds for dynamic and partially dynamic graph HAC with complete linkage, weighted average linkage, and average linkage.	$O(k \log^6 n)$	2022

B. Minimum Spanning Tree Maintenance

The Henzinger and King in [11] presented the first deterministic and fully dynamic algorithm for MST maintenance with an amortized cost of $O(n^{1/3} \log n)$ per update operation, making it significantly faster than previous results for updating the MST with each change in the graph structure, depending on sparsification technique to reduce running time. The paper also utilizes ET-trees, a specific type of B-tree (binary trees), to manage the ET-sequences generated from the graph's trees, facilitating quick splits and joins within the forest that represents the MST, thus supporting the algorithm's efficiency. The algorithm's performance is measured in amortized time, which means it shows the average time per operation over a sequence of operations, but this does not guarantee the time for each individual operation to be consistently low.

Ribeiro and Taso in [20] introduced an online, fully dynamic algorithm for updating minimum spanning trees in graphs when edge weights change, showing its efficiency through numerical experiments and using a simple data structure called DR-trees and DRD-trees for dynamic trees.

RD-trees (rooted reversed dynamic trees) help manage tree connections and disconnections quickly by keeping track of parent-child relationships and the weight of their connections. However, some operations like linking two parts of a tree can take longer because they need extra steps. DRD-trees replaced the DR-trees by implementing doubly-linked trees. The new DRD-trees improve on RD-trees by adding a way to easily see all the children of a node, making it easier to figure out how to reconnect parts of a tree after a disconnection, aiming for quicker and more efficient tree management.

The proposed framework for implementing dynamic algorithms can handle changes in edge weights, proving through extensive experimental analysis that these methods perform well even under challenging update sequences. Experimental results on structured updates and large instances demonstrate that the proposed algorithms, especially variants using DRD-trees, outperform existing solutions in terms of computation times, making them suitable for graphs with complex structure.

In [7], Miroslaw et al introduced the concept of competitive maintenance of minimum spanning trees in dynamic graphs, where an online algorithm is tasked with choosing an MST after each change in the graph's edge weights. The authors highlight that the number of changes in the MST between rounds is a key complexity parameter and discuss the significance of minimizing these changes. They provide insights into the application of the proposed algorithms in the context of mobile networks, specifically in forming communication structures between network parts using mobile relay stations. The paper also emphasizes the relevance of their model in scenarios where changes in the minimum spanning tree are costly, such as in networks where trees are used as a routing or overlay structure. It details the deterministic algorithm, MSTMark, designed to maintain the MST and achieve an optimal competitive ratio of O(n2) for any graph with n vertices. It also presents the randomized algorithm, RandMST, which achieves an expected competitive ratio of O(n log n) for general graphs and O(log n) for planar graphs. The analysis of the algorithms involves the concepts of fixed components, edge sets, and realms in different layers, and the interaction between these components.

The Dynamic Spanning Tree with Mobile Sink (DSTMS) routing algorithm, proposed by Qian et al. in [23], aims to enhance data transmission routes in wireless sensor networks by using a mobile sink. The algorithm creates a transmission framework with several layers using a dynamic MST based on the Low Energy Adaptive Clustering Hierarchy Protocol (LEACH). The process entails choosing meeting spots based on the motion parameters of the

mobile sink to establish a dynamic meeting layer that directs the hierarchical transmission of DSTMS. In order to assess the expenses associated with transmission routes in the MST, a weight factor is incorporated. This component is determined using an energy-efficient function that takes into account variables such as transmission energy usage and the distribution of remaining energy. The DSTMS algorithm is designed to minimize the amount of energy consumed by the network due to frequent updates of the mobile sink's location. It also intends to reduce congestion in data transmission by employing a dynamic hierarchical transmission scheme. The algorithm's usefulness in extending network lifespan, distributing network traffic evenly, and minimizing wireless transmission collisions is demonstrated through simulations, highlighting its potential to enhance the performance of wireless sensor networks with mobile sinks.

In [16], Mao et al. discussed an algorithm for efficiently maintaining the minimum spanning tree (MST) in dynamic graphs. The algorithm aims to update the MST when the graph undergoes structural changes, such as adding or removing edges and vertices. The study presents a detailed analysis of the algorithm's time complexity and correctness, providing theoretical proofs to support its effectiveness. Additionally, the experimental results compare the proposed algorithm with Kruskal's algorithm in terms of runtime efficiency, evaluating its performance in scenarios involving supergraphs and subgraphs.

The experimental results for the supergraph maintenance algorithm demonstrate its superiority over Kruskal's algorithm in efficiently updating the MST. The proposed algorithm consistently outperformed Kruskal's algorithm in terms of runtime, particularly as the number of added edges increased. Additionally, the algorithm showed better performance in denser graphs compared to sparse ones, indicating its effectiveness in handling larger and more complex graph structures.

Furthermore, the experimental analysis for the subgraph maintenance algorithm is anticipated to provide valuable insights into the algorithm's performance in updating the minimum spanning tree for subgraphs. The results are expected to showcase the algorithm's efficiency in handling structural changes within the graph, shedding light on its practical applicability and effectiveness in dynamic graph maintenance scenarios.

Table 2: Summary Of Major Research Contributions in The Randomized Dynamic MST.

Paper Title	Authors	Main Contributions	Time Complexity	Year
Fully-Dynamic Minimum Spanning Forest with Improved Worst-Case Update Time	Christian Wulff-Nilsen	Las Vegas algorithm for MSF with improved update time	$O(n^{1/2-c})$	2017
Dynamic Minimum Spanning Forest with Subpolynomial Worst-Case Update Time	Danupon Nanongkai, Thatchaphol Saranurak, Christian Wulff-Nilsen	Las Vegas algorithm for dynamic MSF with subpolynomial update time	$O(n^{o(1)})$	2017
DynamicSpanningForestwith Worst-Case Update Time: Adaptive, Las Vegas, and $O(n^{1/2-\epsilon})$ Time	Danupon Nanongkai, Thatchaphol Saranurak	Monte Carlo and Las Vegas algorithms for dynamic MSF	$O(n^{0.4+o(1)})$ (Monte Carlo), $O(n^{0.49306})$ (Las Vegas)	2017
On the Dynamic Maintenance of Spanning Tree	Isha Singh, Bharti Sharma, Awadhesh Kumar Singh	Heuristic method for secure and dynamic maintenance of spanning trees in wireless networks	Not specified	2016
Fully Retroactive Minimum Spanning Tree Problem	Jose Wagner de Andrade Junior, Rodrigo Duarte Seabra	Retroactive MST maintenance using square root technique and link-cut tree	$O(\sqrt{m}\ log(\|V(G)\|))$	2022

Constant-Time Dynamic Weight Approximation for Minimum Spanning Forest	Monika Henzinger, Pan Peng	(1+ϵ) approximation of MSF weight with deterministic and Monte Carlo approaches	$O(\log W/\epsilon^4)$	2021

III. Randomized Fully Dynamic Graph Algorithms

Wulff-Nilsen in [24] addressed the problem of maintaining an MSF in a dynamic graph where edge insertions and deletions occur. The suggested solution is a Las Vegas algorithm that achieves an anticipated worst-case update time of ($O(n^{1/2-c})$) for some constant ($c > 0$), surpassing the previous ($O(\sqrt{n})$) bound established by Eppstein et al. This advancement is accomplished by converting from a fully dynamic to a decremental MSF and utilizing innovative methods such as monitoring low-conductance cuts in dynamic graphs. The essential elements of the strategy include:

- Ensuring the presence of clusters with low conductance and utilizing expander graphs to effectively handle the ever-changing nature of the graph.
- An iterative clustering algorithm and a hierarchical data structure called a top tree are used to effectively manage the multi-scale fusion (MSF) process.
- A reduction approach is proposed to transform a fully dynamic problem into a decremental MSF problem, while guaranteeing worst-case time preservation.
- The introduction of a Monte Carlo variant of the structure that can handle updates in worst-case time ($O(n^{1/2-c})$) with high probability.

The study additionally provides a comprehensive description of the execution and evaluation of the data structure, encompassing preliminary procedures, managing updates, and efficiently handling interconnected clusters. The text explores the utilization of auxiliary operations to uphold and modify specific areas of the graph, guaranteeing the accurate preservation of the MSF despite any dynamic alterations to the graph.

In [19] Nanongkai et al. introduced a Las Vegas algorithm specifically designed to dynamically MSF in a graph with n nodes, while experiencing edge insertions and deletions. This technique dramatically enhances performance compared to prior efforts by Wulff-Nilsen, Nanongkai and Saranurak, as it ensures a worst-case update time of O(n o(1)) with high probability. The main progress comes from enhancing the speed of updating the removal of low-conductance cuts in an expander and offering a new method for preserving a minimum spanning forest in graphs with a small number of edges.

The algorithm's structure is based on a shared framework utilized by earlier studies, which involves merging and improving upon their concepts. This system comprises three primary elements: expansion decomposition, expander pruning, and the preservation of MSF/SF in extremely sparse graphs. The enhancements are outlined in the dynamic expander pruning and the approach for managing graphs with few non-tree edges, which utilize flow-based strategies to attain quicker and more effective outcomes. The model comprises three primary components:

- Growth Decomposition: This component breaks down the input graph into multiple expanders and a residual section with a small number of edges (o(n)). An expander is a subgraph with a high degree of connectivity that enables efficient algorithmic operations.
- Expander Pruning: This component is responsible for preserving the MSF in the expanders that were found in the first component. It achieves this by systematically deleting low-conductance cuts in an expander that is experiencing edge deletions in a step-by-step manner. Low-conductance cuts refer to partitions of the graph that have a limited number of edges passing between the partitions, indicating a low level of connectedness.
- Dynamic MSF/SF on Ultra-Sparse Graphs: This component ensures the continuous maintenance of the MSF in the remaining portion of the graph that is derived from the expansion decomposition. This section exhibits a limited number of edges, hence facilitating its management. The authors propose a novel method to address this

issue by transforming the problem on graphs with a small number of non-tree edges into graphs with even fewer edges and nodes, enabling the iterative implementation of their strategies.

Nanongkai et al. introduced two algorithms in [18] to manage dynamic MSF in dynamic graphs. These techniques provide enhancements compared to the traditional worst-case update time bound of O(n) and are specifically developed to counter adaptive adversaries. The first algorithm is a Monte Carlo algorithm with a worst-case update time of $O(n^{0.4+o(1)})$. The second algorithm is a Las Vegas algorithm with a worst-case update time of $O(n^{0.49306})$. Both algorithms utilize graph decomposition techniques to manage subgraphs that have significant expansion or sparsity. They combine new developments in static flow computations with traditional dynamic graph algorithm techniques.

The paper's technical core explains the process of breaking down graphs into high-expansion and sparse subgraphs, which enables the effective maintenance of MSF. The fundamental concept behind the Monte Carlo approach involves the preservation of a spanning forest within each high-expansion component. This is achieved using a combination of random sampling and sparse recovery techniques. The Las Vegas algorithm enhances its functionality by incorporating error detection and correction mechanisms, hence ensuring accuracy and eliminating any potential faults. This requires a more sophisticated approach to managing graph deconstruction and update procedures, guaranteeing that time limits for the worst-case scenario are achieved with a high level of certainty.

The paper provides comprehensive information on the tools and techniques employed in these algorithms, including the augmented Euler Tour Tree, cut recovery trees, and 2-dimensional ET trees. It explores the mathematical underpinnings and logical demonstrations of the algorithms' accuracy and effectiveness, while also examining the compromises and variables associated with their execution. This article makes a substantial contribution to the subject of dynamic graph algorithms by introducing innovative and fast techniques for preserving dynamic MSF. These techniques provide strong theoretical guarantees, even when dealing with adaptive adversaries.

Singh et all in [21] presented a centralized heuristic method specifically developed to ensure the secure and dynamic maintenance of spanning trees in wireless networks. At first, the method creates a minimal spanning tree to depict the network topology, which serves as a fundamental basis for subsequent operations. In order to handle changes in the network's structure while maintaining security, the minimum spanning tree is dynamically reorganized. This ensures that the final spanning tree accurately represents the network's dynamics. While the rearranged structure may not necessarily be a minimal spanning tree, it places a higher emphasis on adaptability and security when dealing with changes in the network. The effectiveness of the proposed method is confirmed by simulations, showcasing its capability to sustain safe and dynamic spanning trees in wireless networks. The paper's key contributions are providing a pragmatic method for preserving spanning trees in wireless networks that can adjust to evolving network conditions while ensuring security.

In [6], Andrade and Rodrigo presented their approach of maintaining retroactive MST. The entirely retroactive MST enables the addition of an edge to a specific time (say t) or the acquisition of the current MST at time t. By employing the square root technique and a data structure link-cut tree, it was feasible to develop an algorithm that amortizes each query in $O(\sqrt{m}\ \lg |V(G)|)$. Here, $|V(G)|$ represents the number of nodes in graph G and m represents the timeline's size. They employed an alternative method to resolve the MST problem, as opposed to conventional algorithms like Prim or Kruskal. This enables them to enhance the ultimate complexity of the algorithm by employing the square root technique. The empirical analysis demonstrated that the proposed algorithm is more efficient than re-executing the standard algorithms, and this performance disparity is exacerbated as the number of nodes in these graphs increases.

Henzinger and Peng in [10] introduced two techniques that are capable of dynamically maintaining a $(1+\epsilon)$-approximation of the weight of an MSF in a graph with n nodes and edge weights ranging from 1 to W, where ϵ is any positive value. The first approach is deterministic and has a worst case update time of $O(W^2 \log W/\epsilon^3)$. This time complexity remains constant if both W and ϵ are constant values. The second approach is a randomized (Monte Carlo) technique that has a worst-case update time of $O(\log W/\epsilon^4)$. Here, W is defined as $O((m)^{1/6}/\log^{2/3} n)$, where m represents the lowest number of edges in the graph for all updates. This algorithm employs randomization and has a high likelihood of success, even when faced with a clever opponent. It also achieves update times that are less than logarithmic, given specific conditions. The paper explores the unresolved issue of whether it is possible to

maintain a (1+ϵ)-approximation of the MSF weight in sub-logarithmic time. It presents algorithms that offer an affirmative solution to this topic.

The paper provided detailed information about the technological advancements of the algorithms, specifically their methods for estimating the number of connected components (CCs) in a graph and utilizing this information to approximate the weight of the Minimum Spanning Forest (MSF) in subgraphs. The deterministic algorithm utilizes local exploration and keeps track of counts for small connected components (CCs), whereas the randomized algorithm takes advantage of property testing techniques and dynamically samples non isolated vertices. Both techniques demonstrate substantial enhancements in update times when compared to current methods across a wide range of parameters.

Furthermore, the paper presented evidence and elaborate explanations of the algorithms, encompassing preprocessing procedures, update management, and the utilization of self-loops for synchronization in the randomized method. The text further explores the concept of lower bounds, demonstrating that for certain situations, any deterministic or randomized data structure that dynamically tracks the (1+ϵ)-approximate weight of an MSF necessitates more than constant time per operation. The given evidence utilizes reductions from established challenges in dynamic connection to create these lower limits.

IV. Conclusion

In conclusion, this paper provides a comprehensive overview of the significant advancements in maintaining Minimum Spanning Trees (MST) in fully dynamic graphs. The paper highlights the critical role of dynamic graph processing in various real-world applications, such as social network analysis, traffic forecasting, and brain functionality modeling. The integration of topological and temporal information, along with the development of optimized data structures and concurrent algorithms, has pushed the boundaries of efficiency, applicability, and performance in dynamic graph maintenance.

Key contributions from researchers like Eppstein et al., Holm et al., Henzinger and King, and Ribeiro and Toso have set new benchmarks in the field, demonstrating substantial improvements in time complexity and operational efficiency. The exploration of competitive maintenance algorithms and dynamic graph neural networks further illustrates the broad applicability and potential of these advancements. The paper underscores the importance of continued research into leveraging new memory technologies and scalable visualization techniques to handle increasingly large and complex datasets. The collective progress made in dynamic MST maintenance not only enhances our understanding of dynamic graph algorithms but also paves the way for future innovations in this critical area.

Acknowledgment

The author extends his appreciation to the Deanship of Scientific Research at Northern Border University, Arar, KSA for funding this research work through the project number "NBU-FFR-2024-1580-06".

References

[1] Sebastien Adam and Cl´ ement Chatelain. Dynamic graph representation´ learning with neural networks: A survey. arXiv.org, 2023.

[2] McCrabb Andrew, Nigatu Hellina, Getachew Absalat, and Bertacco Valeria. Dygraph. 2022.

[3] Muhammad A. Awad, Saman Ashkiani, Serban D. Porumbescu, and John D. Owens. Dynamic graphs on the gpu. 2020.

[4] Michael Burch, Kiet Bennema ten Brinke, Adrien Castella, Ghassen Karray Sebastiaan Peters, Vasil Shteriyanov, and Rinse Vlasvinkel. Dynamic graph exploration by interactively linked node-link diagrams and matrix visualizations. Visual Computing for Industry, Biomedicine, and Art, 2021.

[5] Bapi Chatterjee, Sathya Peri, and Muktikanta Sa. Dynamic graph operations: A consistent non-blocking approach. arXiv: Distributed, Parallel, and Cluster Computing, 2020.

[6] Jose Wagner de Andrade Junior and Rodrigo Duarte Seabra. Fully retroactive minimum spanning tree problem. The Computer Journal, 65(4):973–982, 2022.

[7] Miroslaw Dynia, Miroslaw Korzeniowski, and Jarosław Kutyłowski. Competitive maintenance of minimum spanning trees in dynamic graphs. In International Conference on Current Trends in Theory and Practice of Computer Science, pages 260–271. Springer, 2007.

[8] David Eppstein, Giuseppe Italiano, Roberto Tamassia, Robert Tarjan, Je Ery Westbrook, and Moti Yung. Maintenance of a minimum spanning forest in a dynamic plane graph. Journal of the ACM (JACM), 39(1):81– 97, 1992.

[9] Leonidas Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi. ACM transactions on graphics (TOG), 4(2):74–123, 1985.

[10] Monika Henzinger and Pan Peng. Constant-time dynamic weight approximation for minimum spanning forest. Information and Computation, 281:104805, 2021.

[11] Monika R Henzinger and Valerie King. Maintaining minimum spanning trees in dynamic graphs. In Automata, Languages and Programming: 24th International Colloquium, ICALP'97 Bologna, Italy, July 7–11, 1997 Proceedings 24, pages 594–604. Springer, 1997.

[12] Jacob Holm, Kristian De Lichtenberg, and Mikkel Thorup. Polylogarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. Journal of the ACM (JACM), 48(4):723–760, 2001.

[13] Jacob Holm, Eva Rotenberg, and Christian Wulff-Nilsen. Faster fullydynamic minimum spanning forest. In Algorithms-ESA 2015: 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings, pages 742–753. Springer, 2015.

[14] Bo Jiang, Yuming Huang, Ashkan Panahi, Yiyi Yu, Hamid Krim, and Spencer L. Smith. Dynamic graph learning: A structure-driven approach. 2021.

[15] Xiaona Li, Zhu Wang, Xindong Chen, Bin Guo, and Zhiwen Yu. A hybrid continuous-time dynamic graph representation learning model by exploring both temporal and repetitive information. ACM Transactions on Knowledge Discovery From Data, 2023.

[16] Mao Luo, Huigang Qin, Xinyun Wu, Caiquan Xiong, Dahai Xia, and Yuanzhi Ke. Efficient maintenance of minimum spanning trees in dynamic weighted undirected graphs. Mathematics, 12(7):1021, 2024.

[17] Yao Ma, Ziyi Guo, Zhaochun Ren, Eric Zhao, Jiliang Tang, and Dawei Yin. Dynamic graph neural networks. 2018.

[18] Danupon Nanongkai and Thatchaphol Saranurak. Dynamic spanning forest with worst-case update time: adaptive, las vegas, and o (n1/2-ε)time. In Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, pages 1122–1129, 2017.

[19] Danupon Nanongkai, Thatchaphol Saranurak, and Christian WulffNilsen. Dynamic minimum spanning forest with subpolynomial worstcase update time. In 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), pages 950–961. IEEE, 2017.

[20] Celso C Ribeiro and Rodrigo F Toso. Experimental analysis of algorithms for updating minimum spanning trees on graphs subject to changes on edge weights. In Experimental Algorithms: 6th International Workshop, WEA 2007, Rome, Italy, June 6-8, 2007. Proceedings 6, pages 393–405. Springer, 2007.

[21] Isha Singh, Bharti Sharma, and Awadhesh Kumar Singh. On the dynamic maintenance of spanning tree. In Proceedings of the International Congress on Information and Communication Technology: ICICT 2015, Volume 2, pages 213–222. Springer, 2016.

[22] Tom Tseng, Laxman Dhulipala, and Julian Shun. Parallel batchdynamic minimum spanning forest and the efficiency of dynamic agglomerative graph clustering. In Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures, pages 233– 245, 2022.

[23] Qian Wei, Ke Bai, and Lin Zhou. An improved approach for wireless sensor networks with mobile sink using dynamic minimum spanning tree. IEEE Sensors Journal, 22(11):10918–10930, 2022.

[24] Christian Wulff-Nilsen. Fully-dynamic minimum spanning forest with improved worst-case update time. In Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, pages 1130–1143, 2017.