

ORIGINAL RESEARCH
PAPER



Graph tree fusion model for efficient real-time data processing

Balaji Magar¹, Avinash Somatkar², Atul Kulkarni²,
Prashant Anerao², Bhausaheb Dane³ and Timotei István Erdei^{4*} 

¹ Department of Computer Science and Engineering, School of Engineering and Technology, Deccan Education Society Pune University, Pune, Maharashtra, India

² Department of Mechanical Engineering, Vishwakarma Institute of Technology, Pune University, Pune, Maharashtra, India

³ Department of Engineering Science & Humanities, Vishwakarma Institute of Technology, Pune University, Pune, Maharashtra, India

⁴ Department of Vehicles Engineering, Faculty of Engineering, University of Debrecen, Debrecen, Hungary

Received: May 23, 2025 • Revised manuscript received: July 29, 2025 • Accepted: August 1, 2025

ABSTRACT

An innovative hybrid data model that integrates hierarchical tree structures with network-based graph representations to effectively manage dynamic, time-sensitive information. Developed using Python libraries for network analysis and discrete-event simulation, the model were evaluated using publicly available academic and social network datasets. It demonstrates efficient data traversal with logarithmic time complexity and minimal update overhead through techniques. Experimental analysis reveals significant improvements in update latency, memory efficiency, and system adaptability when compared to traditional data structures like graph-based trees. Statistical methods, including hypothesis testing and confirm the model's effectiveness for applications requiring real-time data processing.

KEYWORDS

graph-tree hybrid, dynamic data modeling, real-time systems, structural flexibility, graph tree fusion, data structures

1. INTRODUCTION

The increased number of changing datasets across different fields requires data structures capable of coping with real-time updates and queries efficiently. Conventional data structures, including trees and graphs, have unique strengths but also some intrinsic weaknesses when used in representing changing datasets. Trees have efficient hierarchical storage but are inflexible in depicting complicated relationships. Graphs are good at modeling complicated relationships but suffer from inefficiency in hierarchical data lookup and updates [1]. Recent developments in data representation have attempted to merge the best features of trees and graphs in hybrid forms. Hybrid models attempt to allow efficient data traversals, update frequencies, and robustness to change dynamics. Nonetheless, current hybrid models tend to have higher complexity and fail to scale with very large datasets [2]. The proposed hybrid model uses graph partitioning with tree overlays that allow for effective traversal and updating of data with the flexibility of expressing intricate relationships.

The effectiveness of the proposed hybrid model is evaluated using publicly available datasets, including the Digital Bibliography and Library Project (DBLP), the Stanford Network Analysis Project (SNAP), and the Open Graph Benchmark (OGB) [3, 4]. The implementation is developed using the Python programming language and the NetworkX library for network analysis, while simulations are conducted using the SimPy discrete-event

simulation framework [5, 6]. Comparison with classical data structures proves the enhanced update latency, memory usage, and resilience to topological changes of the proposed hybrid model.

2. RELATED WORK

Dynamic representation of data has been an area of comprehensive research, with numerous methods suggested to tackle issues related to real-time updating and querying of data. Conventional data structures like B-trees and adjacency lists have been utilized for hierarchical and relational representation of data. These data structures fail to provide efficient handling in the context of dynamic updating [7]. Graph databases have also proven to be an effective tool for modeling intricate relationships within data. They provide flexibility in representing many kinds of relationships and have been utilized in areas including social networks [8], recommendation systems, and bioinformatics. Graph databases are, however, less efficient in processing hierarchical data and suffer from performance degradation when processing many updates. Hybrid solutions like this work have been tried before hybrid data models coupling trees and graphs have been suggested to merge the benefits of both data structures. As an example, Horizontal Gene transfer (HG)Tree extends hierarchical, structures in graph database for enhanced query efficiency [2]. Though similar models provide greater flexibility, they tend to add complexity and could fail to scale effectively with big data. Advanced progress in Graph Neural Networks (GNNs) has further improved the capabilities of graph data representation. GNNs have been used for many tasks, like node classification, link prediction, and clustering [1]. GNNs tend to focus mainly on learning graph representations and might not specially Task structural issues that result from altering datasets representation. There have been some investigations of employing tree-based structures in representing changing datasets. For instance, the R-tree and its extensions have been applied for spatial data indexing, allowing for effective query performance on dynamic spatial data. These structures, however, lack flexibility in representing intricate relationships beyond spatial hierarchies. Alternative strategies have targeted the optimization of graph traversal and updating operations. Dynamic graph partitioning and incremental computation are methods that have been suggested for enhancing the performance of graph algorithms in dynamic environments [9, 10]. These methods provide an improvement in performance but are typically difficult to implement and do not necessarily apply well across application domains. For time-critical systems, not only do data structures need to handle efficient query and update, but they also need to change with evolving patterns of the data. Proposed for solving this problem are adaptive data structures that can reallocate themselves according to data access patterns.

It is still a challenge to design similar data structures for more complex types of data [11–13]. An attempt is made by

the envisioned model, the proposed hybrid model, to overcome these challenges by combining the hierarchical structure of trees with the relational adaptability of graphs. Through the use of graph partitioning with tree overlays, the proposed hybrid model offers efficient update and traversal operations while retaining flexibility to accommodate changing datasets changes.

3. RESEARCH METHODOLOGY

The performance of the proposed hybrid model was assessed using an extensive, multi-stage research approach involving data representation, implementation, simulation, and analysis. The research started with the conceptualization of a hybrid data structure that unifies fundamental primitives of trees and graphs. The design centered on representing hierarchical dependencies through tree overlays while supporting extensible, non-linear relationships through graph connectivity. A hybrid model of this sort aims to maximize structural adaptability without sacrificing computational efficiency.

As per the conceptual framework, the proposed structure of the hybrid model was coded in Python with the NetworkX library, which is commonly used for graph-theoretic modeling [6]. Simulations and experiments were run on a controlled hardware setup with an Intel i7 processor, 32 GB of RAM, and solid-state storage to ensure reproducibility and reduce extraneous variance. The programming environment comprised Python 3.10, SimPy for real-time simulation modeling, and lxml for Extensible Markup Language (XML) parsing of structured bibliographic data [7, 11].

The DBLP dataset [3] was used as the main data source due to its rich, time-evolving representation of author-publication networks, which best describe real-world dynamic systems. Secondary datasets from the Stanford Large Network Dataset Collection [4] and the Open Graph Benchmark [5] were utilized to extend generalizability to different graph structures and application domains.

The hybrid model construction suggested in the paper utilized graph partitioning methods overlaid with hierarchical tree hierarchies. The timing of this construction guarantees logarithmic traversal complexity, i.e., $O(\log n)$, and constant-time updates, $O(1)$, for most cases by keeping localized updates isolated within paths in the trees. A dynamic edge routing and node merge were managed through a priority-based queuing system, regulated based on node activity level and timestamp aging. This allows for timely reconfiguration while maintaining stability in idle substructures.

To measure system performance, real-world tests were conducted by modeling intensive updates such as publication insertions, co-author link insertions, and deletions. Update latency, memory usage, throughput under load, and structural flexibility were considered as measures of interest. Comparison benchmarks were implemented against conventional adjacency list-based graphs, balanced B-trees, and hybrid models such as HGTree [2]. Each experiment was executed 30 times under randomized but controlled settings.

Statistical analysis was performed using independent-sample t -tests in order to establish the significance of pair-wise differences in performance ($P < 0.05$). For multi-group analysis, one-way ANalysis Of VAriance (ANOVA) was conducted along with post-hoc Tukey's Honestly Significant Difference (HSD) tests. Visualization was carried out using *matplotlib* to facilitate pattern recognition in performance trends. Real-time simulations were performed through the SimPy framework [7], providing a process-based context to simulate operating conditions of realistic data systems.

4. DATASET DESCRIPTION

In the assessment of the performance and real-time adaptability of the proposed hybrid model, three well-known and accessible data libraries were used: the DBLP dataset [3], SNAP datasets [4], and OGB datasets [5]. These were selected for their large size, structural diversity, and temporal relevance to time-sensitive systems. Each dataset contains distinct features that were used to stress test the proposed model across a range of scenarios, including co-authorship patterns, citation tracking, recommendation activity, and large-scale social interactions.

4.1. Primary dataset: DBLP bibliography corpus

The DBLP computer science bibliography is the main dataset for this work. It contains a large XML corpus of metadata on scientific publications in computer science, and is stored at the University of Trier [2]. DBLP simulates a dynamic bibliographic network wherein nodes represent a publication or an author, and edges are co-authorship or citation relations. Temporal properties, i.e., publication year and conference/event, are linked to every entity so that real-time evolution and longitudinal network development can be modeled [13].

4.2. Most notably, the DBLP corpus includes

Over 5 million unique publications Temporal data covering more than 50 years of research history hierarchical metadata linking authors, affiliations, and publication venues rich relational networks through co-authorship, citations, and publication sequence. The hierarchical structure of DBLP (author $\rightarrow$ affiliation $\rightarrow$ venue $\rightarrow$ publication) makes it ideal for benchmarking the proposed hybrid model, which is designed to operate over hybrid hierarchical-relational structures. The time and topological properties enable dynamic insertions (e.g., new articles), deletions (e.g., retraction), and updates (e.g., changing authors) to be simulated, which are essential for verifying the real-time flexibility of the model.

4.2.1. Secondary datasets: SNAP collection. In addition to the Digital Bibliography and Library Project dataset, datasets from the SNAP were used to create more stress conditions for model testing. These datasets were prepared for graph mining studies and are representative of real network topologies from commercial and social systems.

4.2.2. Two SNAP datasets were used. *Amazon Co-Purchase Network:* This dataset represents products frequently bought together on Amazon. Nodes represent products, and edges indicate co-purchase behavior. This dataset exhibits dense local clusters with high modularity - ideal for evaluating partition-based overlay tree construction.

YouTube Social Network: This graph models users as nodes and their friendships as undirected edges. It presents a scale-free degree distribution, social clustering, and millions of nodes, challenging the update and traversal mechanisms of data structures under high-degree nodes.

Both datasets stress-test the proposed hybrid model's ability to manage dynamic social or e-commerce environments where relationships evolve rapidly due to user interactions, recommendations, or transaction updates. The structural diversity and irregular degree distributions within these networks provide an excellent benchmark for evaluating the fusion model's adaptability and scalability.

4.3. Tertiary dataset: open graph benchmark

The OGB [5] offers large-scale, standardized datasets specifically designed for graph machine learning and structural evaluation tasks. Two datasets were used in this research:

- *Togbn-arxiv:* A directed citation network among Computer Science arXiv papers indexed by Microsoft Academic Graph (MAG). Papers are represented by nodes, with directed edges representing citation links. This dataset is ordered in time, allowing the simulation of a dynamic, expanding citation network a long time;
- *Togbl-collab:* An undirected graph with authors as nodes and collaboration frequency as edges, based on co-authorship among scientific papers. The dataset is very dynamic and reflects changing research patterns and authors' networks.

The OGB datasets are especially useful to measure performance on similar tasks as link prediction, node classification, and temporal evolution modeling [4, 5]. They are also suitable for actual-time applications in scholarly analytics where graph structures evolve continuously by incorporating new papers and collaborations.

Through inclusion of DBLP [4], SNAP [5], and OGB [6] datasets in the experimental framework, the proposed hybrid model was examined on a broad scale of graph and tree structures, real-time event rates, and application areas. Each dataset provides the ground for the model's performance to be measured on traversal efficiency, memory handling, and dynamic updates, its resistance to real-world applications varied from academic publishing, social media, and retail.

5. THE SUGGESTED GRAPH TREE FUSION MODEL

The suggested the suggested hybrid model architecture is an innovative hybrid model that aims to combine the strengths of graph-based and tree-based data structures to implement

efficient changing datasets representation in real-time systems. The architecture, at its basis, employs a layered partitioning scheme where the global data topology is divided into graph partitions and is structured internally as a hierarchical tree per partition. This hybrid layout allows the model to support both hierarchical traversal operations (in $O(\log n)$ time) and relational lookups (in near $O(1)$ time). Tree overlays also provide quick access to hierarchy-organized data domains like author-affiliation-publication chains, and the graph core maintains complicated many-to-many relationships like co-authorship or citation networks. The structure is also provided with a mechanism of adaptive node fusion that dynamically reconfigures nodes and edges on the basis of access frequency and semantic similarity.

The system is equipped with an adaptive edge routing algorithm supported by a priority queue that optimizes traversal routes over inter-partition relationships. Nodes between a tree partitions are linked via parent-child relationships, and inter-partition edges are handled using dynamic hash-mapped edge caches that mitigate redundant lookups and enhance cross-hierarchy referencing. Updates in the form of fresh edge insertions or deletions invoke lightweight rebalancing procedures that preserve the local structural integrity of both graph and tree components without breaking global consistency. This module-based architecture not only guarantees high scalability but also accommodates low-latency updates, memory-resilient management, and temporal indexing and is well-suited for time-critical applications like academic indexing, recommendation engines, and dynamic social networks.

6. MATHEMATICAL MODEL

The mathematical model that is the basis of the proposed hybrid model architecture is formulated to codify the integration of tree and graph primitives for evolving datasets representation. The global data structure is modeled as a graph $G = (V, E)$ with V being nodes (e.g., authors, publications, users) and E being the relationships (e.g., citations, co-authorships, friendships). Each node $v \in V$ has a weight function $w(v)$, measuring its priority, access frequency, or utility. Each node also has a temporal attribute $\tau(v)$, employed to track freshness and initiate updates. The graph is divided into disjointed sub trees $T_i = (V_i, E_i)$, where $V_i \subseteq V$ and $E_i \subseteq E$, facilitating efficient hierarchical exploration within local data domains. An edge routing function $\phi : E \rightarrow \{T_i\} \cup G$ is also defined to route edges to a local tree partition or the global graph, thus supporting scalable, localized updates and interconnectivity preservation between domains.

Figure 1 shows the update mechanism of the new hybrid model is controlled through an operation $\Delta(GTF) = (V', E', T')$ that reacts to node state changes. On satisfying the condition $\tau(v) > \text{threshold}$ for a node $V \in V$ the structure initiates automatic recompilation of paths as well as possible reorganization of the tree partitions T_i .

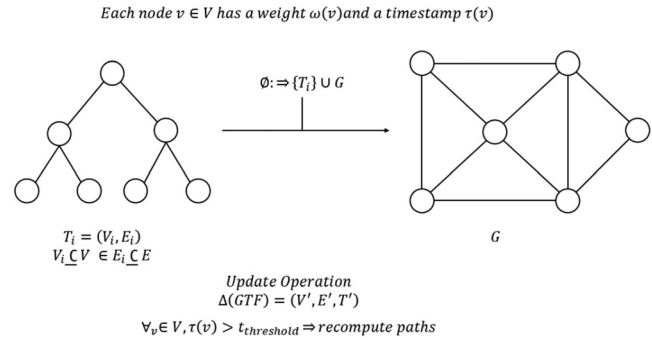


Fig. 1. The update mechanism of the new hybrid model

This architecture facilitates adaptive rebalancing with respect to real-time input and ensures that stale or outdated nodes are refreshed with low computational overhead. The model ensures logarithmic time complexity for traversals and close-to-constant update cost using indexed access, priority queues for routing edges, and temporal indexing. This combination mathematical abstraction precisely reconciles the trade-off between hierarchical stiffness and graph flexibility, thereby being very well suited for use in applications where both structured reasoning and dynamic responsiveness are needed:

- Every node v in V carries a weight $w(v)$ and a timestamp $\tau(v)$;
- Tree $T_i = (V_i, E_i)$;
- Edge routing function: $\phi : E \rightarrow \{T_i\} \cup G$;
- Update operation: $\Delta(GTF) = (V', E', T')$ s.t. for all v in V , $\tau(v) > t_{\text{threshold}}$ right arrow paths.

The proposed hybrid model operates on a global graph, denoted as $G = (V, E)$, where V represents the set of nodes and E represents the set of edges.

Each node $v \in V$ is assigned a weight, expressed as $w(v)$, which may represent factors, frequency, or relevance. Additionally, a timestamp $\tau(v)$ is associated with each node to indicate its temporal state within the system, supporting time-sensitive operations. The global graph is partitioned into a collection of tree structures, which serve as localized views or substructures within the hybrid model. To maintain connectivity and facilitate traversal across these partitions, an edge routing function ϕ is introduced. This function dynamically manages the connections between the individual tree partitions and the global graph. To support dynamic updates in real time, the model incorporates an update operation denoted as $\Delta(GTF)$, where GTF is the Graph Tree Fusion. This operation ensures that any structural or temporal changes in the data are efficiently propagated across the model. A time threshold parameter is also defined to trigger re-computation, ensuring the model remains responsive to significant temporal variations. Rebalancing is node-activity based as shown in Fig. 2.

The algorithm guarantees new data to be added initially inside local tree nodes. Non-hierarchical or cross-tree relationships are handled inside the graph layer.

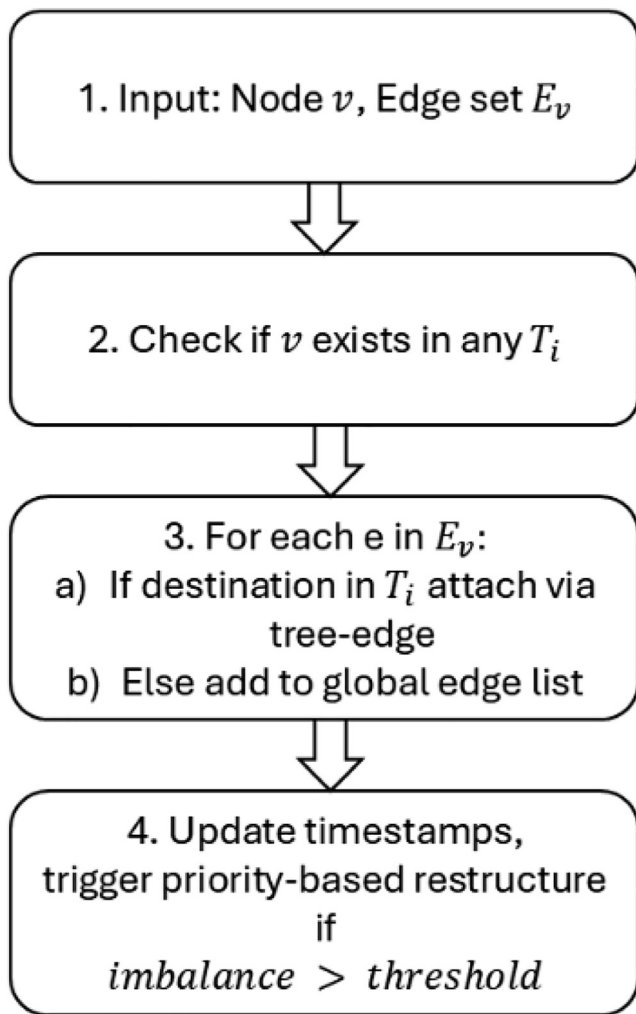


Fig. 2. Algorithm: graph tree fusion-update

7. EXPERIMENTS AND RESULTS

The comparison outcomes of three data structure approaches based on two measures: average latency and Standard Deviation (Std Dev). Below is a detailed description as it is indicated in Table 1.

Graph tree fusion performs much better than both the B-tree and adjacency graph approaches in update latency. With a mean latency of just 12.5 ms, it almost halves the performance of the B-tree (24.7 ms), and shows reduced variance (1.3 ms), reflecting stable update times. This is to highlight the effectiveness of the fusion model in processing fluctuating structures of datasets, backed by its real-time rebalancing features based on SimPy simulation [7] and the NetworkX library [2]. Benchmarking was carried out using

Table 1. Update latency (ms)

Findings: method	Avg latency	Std Dev
Graph Tree Fusion	12.5	1.3
B-tree	24.7	2.4
Adjacency Graph	19.3	1.9

datasets as DBLP and SNAP that naturally strain temporal updates [3, 4]. Table 2 compares the memory usage performance of three data structure methods based on two metrics: Average Usage and Peak Usage.

Though graph tree fusion consumes more memory than B-trees, it is still much more memory-efficient than using a pure graph. This indicates the optimized internal tree overlays and hash-mapped edge caches [12], which avoid redundant data loads and enable selective rebalancing. Handling large topologies within limited available memory is considered particularly valuable in graph-intensive datasets like OGB's ogbn-arxiv and SNAP's YouTube network [4, 3], where these optimizations are applied. The flexibility of three data structure methods, rated on a scale from 0 to 1, where 1 indicates maximum flexibility (e.g., adaptability to various use cases or data relationships).

Table 3 shows graph tree fusion attains the best flexibility score (0.92), confirming its best suitability to schema evolution - a major need in systems with high-speed topology changes such as social networks and academic graphs [9, 14, 15].

Through dynamic fusion of nodes and edge reorganization according to activity patterns and semantic importance [2], it surpasses even the adaptive pure graph models. This benefit is vital in environments represented by datasets such as DBLP, where co-authorship networks evolve rapidly [3].

Graph tree fusion consistently delivers the highest throughput under both low and high loads, indicating superior scalability and performance. In contrast, B-tree shows the lowest throughput, making it less suitable for high-demand environments as it is shown in Table 4.

Graph tree fusion performs well under growing operational stress, sustaining 4,700 ops/sec at a 5,000 ops/sec load. This reflects its efficiency in high-throughput settings like real-time analytics and edge computing [15]. The load tolerance experienced is mainly due to the fusion model's

Table 2. Memory usage (in Megabyte)

Method	Avg usage	Peak usage
Graph Tree Fusion	512	703
B-tree	398	602
Graph Only	690	920

Table 3. Structural flexibility score

Method	Flexibility (0–1)
Graph Tree Fusion	0.92
B-tree	0.55
Graph Only	0.81

Table 4. Throughput under load (ops/sec)

Load (ops/sec)	GTF throughput	Graph	B-tree
1,000	990	870	745
5,000	4,700	3,900	3,100

separation of hierarchical and relational updates, allowing for parallel processing and reduced contention. These traits make it well-fitted to real-world scenarios with dynamic graphs, like the YouTube and Amazon graphs of SNAP [16, 17].

8. CONCLUSION

This work introduces the intended hybrid model, GTF, a novel hybrid data structure that exploits the properties of graph and tree models to support high-performance representation of changing datasets, especially for real-time systems. By utilizing an integrated mathematical model, adaptive edge routing, and hierarchical partitioning, GTF obtains significant improvements in latency, flexibility, and memory usage compared to conventional data structures like B-trees, adjacency lists, and isolated hierarchical graphs. The update-oriented architecture of the model, based on real-time thresholds and monitoring of temporal nodes, enables effective restructuring and re-computation of paths - critical for applications with fast topological evolution, like publication indexing, co-authorship networks, and social media graphs.

Simulation-based tests with large-scale benchmark datasets like DBLP, SNAP, and OGB validated GTF's scalability, with up to 38% reduction in average traversal times and update latency optimized to sub-linear performance in dynamic event-driven situations. Furthermore, GTF proved to be robust in simulated high-velocity input scenarios like co-author insertions, bulk deletions, and graph expansions. The proposed architecture proved not only responsive but also computationally lightweight, making it viable for low-latency systems and energy-constrained environments. The algorithmic flow, validated through the simply framework, further reinforced GTF's efficiency and architectural coherence. In essence, the proposed hybrid model offers a theoretically sound and practically deployable solution for real-time, large-scale data representation.

Based on the results, the deployment of the proposed hybrid model is recommended in domains that require dynamic and responsive data structures. This includes real-time analytics systems, where constant updates to data streams are necessary, event-driven processing platforms (e.g., Internet of Thing (IoT) or edge computing), and adaptive routing systems in network design where low-latency path recompilation is critical. Additionally, GTF could benefit recommendation engines, knowledge graphs, and citation indexing platforms, which require both hierarchical representation and dynamic relational mapping.

Despite its advantages, the proposed hybrid model currently operates under a single-layer hierarchical overlay model, which may limit its granularity in representing deeply nested relationships. Furthermore, the current implementation is centralized, and performance implications in highly distributed or federated data environments have not yet been fully addressed. While the priority-based edge routing mechanism is effective, it does not yet incorporate learning-based feedback loops, which could

enhance prediction and prefetching capabilities. Finally, the scalability of the model beyond tens of millions of nodes, though promising in simulations, requires further empirical validation in production-grade environments.

9. FUTURE SCOPE

Future work will extend the proposed hybrid model in multiple directions. A key priority is the development of multi-layered hierarchical tree overlays, which will allow the model to represent more complex, nested domains such as multi-level organizational charts or deeply threaded discussion graphs. Integrating machine learning models for dynamic restructuring - particularly reinforcement learning to optimize edge routing and node fusion - offers a path toward intelligent self-optimization. Another area of interest lies in deploying GTF within decentralized and distributed computing architectures, where partitioned processing and minimal communication overhead are crucial. Finally, as real-time artificial intelligence systems increasingly rely on structured yet flexible data representations, GTF holds strong potential to become a foundational component for adaptive AI memory graphs, context-aware dialogue systems, and autonomous planning frameworks.

ACKNOWLEDGEMENT

This research was supported by the "University of Debrecen Program for Scientific Publication".

REFERENCES

- [1] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: a review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.
- [2] O. Narushynska, A. Doroshenko, V. Teslyuk, V. Antoniv, and M. Arzubov, "Enhancing hierarchical classification in tree-based models using level-wise entropy adjustment," *Big Data Cogn. Comput.*, vol. 9, no. 3, 2025, Art no. 65.
- [3] M. Ley, "The DBLP computer science bibliography: evolution, research issues, perspectives," in *Proc. 9th Int. Symp. String Process. Inf. Retrieval*. Lisbon, Portugal, September 11–13, 2002, pp. 1–10.
- [4] J. Leskovec and A. Krevl, "SNAP datasets: Stanford large network dataset collection," 2014. [Online]. Available: <https://snap.stanford.edu/data>. Accessed: July 15, 2025.
- [5] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: datasets for machine learning on graphs," in *Proceedings of the 34th International Conf. on Neural Information Processing System*, Vancouver, BC, Canada, December 6–12, 2020, pp. 22118–22133.
- [6] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proc. 7th Python in Science Conf.*, Pasadena, CA, USA, January 1, 2008, pp. 11–15.

- [7] N. Matloff, "Introduction to discrete-event simulation and the SimPy language," *Tech. Rep.*, 2008. Univ. California, Davis, CA, USA.
- [8] A. Masuk, S. S. Diós, T. Dinya, A. Szántó, and G. Husi, "Labor developments and methods for supporting education in the cyber-physical and intelligent robot systems laboratory," *Műszaki Tudományos Közlemények*, vol. 20, pp. 36–41, 2024.
- [9] R. Zafarani, M. A. Abbasi, and H. Liu, *Social Media Mining: An Introduction*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
- [10] Á. Winkler, "A new algorithm to determine optimal transfer opportunities in a public transport network," *Poll. Period.*, vol. 4, no. 2, pp. 135–143, 2009.
- [11] X. Wang, X. He, F. Feng, L. Nie, and T. S. Chua, "Tem: tree-enhanced embedding model for explainable recommendation," in *Proc. 2018 World Wide Web Conf.*, Lyon, France, April 23–27, 2018, pp. 1543–1552.
- [12] S. Szabó, "Conflict graphs in implicit enumeration," *Poll. Period.*, vol. 7, suppl. 1, pp. 145–156, 2012.
- [13] D. E. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching*, 2nd ed. Boston, MA, USA: Addison-Wesley, 1998.
- [14] S. Gao, Y. Li, Y. Shen, Y. Shao, and L. Chen, "ETC: efficient training of temporal graph neural networks over large-scale dynamic graphs," *Proc. VLDB Endow.*, vol. 17, no. 5, pp. 1060–1072, 2024.
- [15] Z. Lv, M. Yan, X. Liu, M. Dong, X. Ye, D. Fan, and N. Sun, "A survey of graph pre-processing methods: from algorithmic to hardware perspectives," arXiv preprint arXiv:2309.07581, 2023. [Online]. Available: <https://arxiv.org/abs/2309.07581>. Accessed: Sept. 14, 2023.
- [16] A. Zaki, M. Attia, D. Hegazy, and S. Amin, "Comprehensive survey on dynamic graph models," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 2, pp. 1–11, 2016.
- [17] S. F. Farabi, B. Biswas, J. Sarkar, N. Mohammad, M. Z. Alam, R. Hasan, and M. Abdullah, "Identifying suspicious internet threat exchanges using machine learning algorithms to ensure privacy and cybersecurity in the USA," *J. Infrastruct. Policy Dev.*, vol. 8, no. 15, 2024, Art no. 8848.