Reducing Unfairness in Distributed Community Detection

Hao Zhang[†], Malith Jayaweera[‡], Bin Ren[§], Yanzhi Wang[‡], Sucheta Soundarajan[†]
[†]Syracuse University {hzhang15, susounda}@syr.edu
[‡]Northeastern University {malithjayaweera.d, yanz.wang}@northeastern.edu
[§]William & Mary {bren}@wm.edu

***Abstract*—Big graph data mining and processing have emerged as a crucial area of study. Distributed graph frameworks are commonly employed to process such big graph data in various applications. These frameworks have proven to be highly effective in improving both the accuracy and efficiency of processing large-scale graph data, but little attention has been paid to the algorithmic fairness of such methods. In this paper, we propose a novel graph reweighting algorithm, `Homophily-Based Graph Reweighting (HBGR)`, which can be used with different distributed community detection frameworks. The findings of our study demonstrate that `HBGR` can significantly enhance the fairness of detected community results, without altering the overall distributed community detection algorithm workflow. Our analysis demonstrates that `HBGR` outperforms traditional performance-based distributed graph data processing frameworks in terms of fairness across 13 real social network datasets. This enhancement enables us to achieve fairness levels that are comparable, or even superior, to those achieved by linear community detection algorithms while maintaining good efficiency performance. Additionally, we examine the causes of unfairness in distributed community detection algorithms and conduct an interpretability analysis of `HBGR`'s improved fairness performance. Finally, we provide a comprehensive evaluation of the trade-offs between efficiency, accuracy, and fairness in distributed community detection algorithms.**

***Index Terms*—big graph data, community detection, big data processing fairness**

I. Introduction

As graph data sizes grow, distributed graph processing algorithms, such as those for community detection, influence maximization, PageRank, and node classification [1], [2], have become popular. However, while such algorithms have made significant improvements in accuracy and efficiency in recent years [3], little attention has been paid to the algorithmic fairness of such techniques [7].

In the field of algorithmic fairness, at a high level, an algorithm is considered ‘fair’ when it does not demonstrate bias against individuals on the basis of their membership in a protected group [15]. In the context of community detection, a fair community structure is one in which each protected group is well represented in each community [12]. Our focus is on community detection, because among the important graph tasks, community detection has become one of the most extensively studied in distributed large graph systems. However, earlier work has shown that while current distributed community structure mining frameworks perform very well with respect to accuracy and efficiency, there is still considerable room for improvement in their fairness performance [5].

Recent research indicates that compared to sequential community detection algorithms, distributed community detection frameworks exacerbate unfairness [5]. The reason for worsened unfairness in distributed community detection systems is the uneven allocation of nodes from different groups to different machines by the distributed system, which leads to higher unfairness in local computations, ultimately causing greater unfairness in the final distributed clustering results [5]. It is important to note that such unfairness can occur even if the algorithm has no knowledge of attributes. This occurs because attributes can be partially reflected in the topology and community structure of the graph, which can influence partition decisions.

In this study, we quantify the causes of unfair community structure in distributed community detection frameworks. Based on this analysis, we propose the novel `Homophily-Based Graph Reweighting (HBGR)` algorithm, which reweights graph edges to change the node distribution across distributed computing machines in order to improve the fairness of community detection. To the best of our knowledge, this is the first graph algorithm for distributed community detection frameworks to address unfairness. Across thirteen datasets, `HBGR` shows excellent performance with respect to fairness.

The main contributions of this paper are as follows:

- We quantify unfairness in distributed community detection algorithm. In particular, in homophillic graphs, graph partitioning methods that improve computing efficiency are more likely to place nodes from the same protected group on the same machine (leading to higher homophily). We propose that distributed community detection techniques' unfair community structures are caused by this sparsity in connections between different groups.
- We introduce a novel graph reweighting algorithm: `Homophily-Based Graph Reweighting (HBGR)`, which can be applied in different distributed community detection frameworks. To the best of our knowledge, `HBGR` is the first algorithm to address unfairness in distributed community detection algorithms, which improves fairness by approximately 30% on average without compromising computational efficiency.
- We demonstrate a trade-off between efficiency, accuracy,

and fairness when applying a distributed community detection framework. Under the condition of sacrificing an average of approximately 10% accuracy, HBGR can improve fairness performance by an average of over 30%, while preserving the current level of computation efficiency.

II. Background and Related Work

Here, we first review state-of-the-art distributed community detection frameworks. We then explain cutting-edge graph partitioning methods used in distributed systems. Finally, we define fairness in community structures.

A. Distributed Community Detection Frameworks

In this work, we consider two community detection algorithms, Louvain [16] and InfoMap [17], and their corresponding distributed frameworks: Vite [4] and RelaxMap [9]. Although there is no single optimal algorithm for community detection (due to the diversity of graph data structures), InfoMap and Louvain are widely recognized as the basis for other approaches and remain highly popular [2].

1) Vite: Louvain groups nodes into distinct clusters in order to optimize modularity, which quantifies the quality of the community structure [16] and Vite performs a distributed computing version of this method [4]. Vite involves two main steps: Louvain iteration and graph reconstruction, which respectively perform local community detection computations as in the sequential Louvain and communicate community structure information across distributed machines.

2) RelaxMap: InfoMap models a random walker to traverse the graph, and then calculates the probability of each node accessing different communities [17]. For distributed InfoMap implementation, RelaxMap searches for new modules of vertices using a lock-free parallel mechanism and the sparsity assumption found in real-world networks to boost efficiency. This approach distributes processor burden evenly across the network [9].

B. Graph Partition Algorithms

Distributed graph applications use graph partitioning methods to split large graphs. For parallel computing, each subgraph is on a separate workstation. A graph partitioning that balances local storage costs over machines and minimize communication costs is known as a balanced graph partitioning.

METIS is a multi-level balanced graph partitioning algorithm [11]. METIS is a fast and precise technique for dividing graphs into partitions that makes the initial graph less dense by merging nodes and edges, streamlines its structure, and reduces its size [10]. METIS minimizes the number of edges across partitions, a critical factor in optimizing the efficiency of distributed graph processing. Balanced graph partition approaches can improve distributed graph computation frameworks. Therefore, in this study, we modify Vite [4] to leverage the METIS graph partitioning technique for graph data pre-processing.

C. Fairness in Community Detection

In general, an algorithm is considered fair if it does not exhibit bias against individuals on the basis of membership in a protected group, such as those based on attributes such as race or gender [15]. This section discusses community detection fairness using group-based terminology. Consider Red and Blue nodes in a social network, reflecting real-world protected groups. For convenience, we employ two groups, although the discussion can be generalized to more than two.

Theoretically, a community structure is considered fair if every protected group is evenly represented in each cluster [6], [12]. The underlying concept of this definition is that every detected cluster should accurately represent the data and encompass its range of variations [12]. Recently, two metrics have been developed to quantify fairness for community structure: Balance-based Community Fairness ($f_{balance}$) [8] and Weighted Imbalance Ratio (WIR) [5].

We mainly use WIR to conduct quantitative analysis on the community structure fairness because it can better evaluate the fairness of the entire community structure. To compute WIR, first, the $imbalance_i$ of community C_i is defined to reflect the balance of different groups of members in the community and it is computed as follows: if the gap between the fraction of Red nodes in community C_i (denoted as $F_R^{C_i}$) and the fraction of Red nodes in entire graph (denoted as F_R) is smaller than a user-defined threshold t, the $imbalance_i$ is set to 0. Otherwise, $imbalance_i$ is the difference between the proportion of the Red group in the community i and the proportion of the Red group in the entire graph, minus t, and can be calculated by Formula (1) [5]:

$$imbalance_i = min(|f_R^{C_i} - (f_R + t)|, |(f_R - t) - f_R^{C_i}|) \quad (1)$$

Because there are multiple protected groups, $imbalance_i$ is calculated for each group and the largest deviation is assigned as the overall community imbalance. Formula (2) [5] calculates the weighted imbalanced ratio (WIR) of the whole community structure after computing $imbalance_i$ for each community. A lower WIR score suggests better community structure fairness.

$$WIR = \sum_{i=1}^{\#\,clusters} (imbalance) \times \frac{\#\,nodes\ in\ cluster\ i}{\#\,nodes\ in\ graph}. \quad (2)$$

III. Proposed Method: Homophily-based Graph Reweighting Algorithm (HBGR)

This section introduces the graph reweighting algorithm, `Homophily-Based Graph Reweighting` (HBGR), which can be applied in different distributed community detection frameworks.

A. Motivation behind the HBGR Algorithm

Research indicates that when distributed computing frameworks utilize graph partitioning algorithms to allocate nodes to machines, in many cases, there is a tendency for similar nodes (such as nodes belonging to the same protected groups) to be placed on the same machine [5]. Critically, this occurs even when the partitioning scheme is *not aware* of node attributes: rather, the attributes are partly reflected in the topology of the graph, which affects partitioning decisions.

This happens in particular on high homophily graphs: those in which there is a strong tendency of nodes to connect to similar nodes (for example, men to men, baseball fans to baseball fans) [14]. Many real-world graphs are known to be homophilic. Because graph partitioning algorithms try to minimize cross-cut edges and a high fraction of edges are between nodes with the same attribute, similar nodes tend to be assigned to the same cluster. Community detection algorithms that run locally over distributed machines follow community detection task goals, such as finding high-density groups. Thus, local clustering computations are more likely to produce unfair results than sequential computations that can access global information.

When nodes are not spread evenly among machines, local community detection computations become biased. Local clustering algorithms perceive the edge density between nodes within the same protected group as higher than it actually is, while the edge density between nodes of different protected groups is lower than it actually is. Local computation struggles to assign nodes from different protected groups to the same community due to sparse edges, leading to increased unfairness. Subsequent cross-machine communication can mitigate but not fully resolve this issue.

Inspired by this observation, we design a method to reweight graph edges so that edges between nodes of different protected groups are prioritized. Balanced graph partitioning algorithms aim to achieve equal edge weights within each subgraph and minimize edge weights between subgraphs. Based on this idea, by assigning higher weights to edges connecting nodes from different groups and lower weights to edges connecting nodes from the same group, we can protect the former from being cut, as higher weights increase cutting costs and balanced graph partitioning algorithms aim to minimize edge cuts, thus encouraging the latter to be cut. Once the graph has been reweighted, the existing graph partitioning method (for weighted graphs) can be used. An overview of the HBGR algorithm is described in Section III-C

.

B. Toy Example of HBGR

Fig. 1 displays the results of Vite-Louvain-based distributed community detection using METIS and HBGR reweighting plus METIS (HBGR-METIS). In the figure, '$+$' indicates that HBGR provides the most protection to that edge, 'o' represents normal protection, and '$-$' indicates a smaller weight from HBGR, causing the partitioning method (METIS) to prioritize cutting that edge. As seen in the illustration, METIS and HBGR prioritize cutting different edges. Although METIS preferentially cuts edges across protected groups, HBGR is more balanced. Consequently, different processors have distinct node and edge distributions, resulting in varied local computation outputs. The final community structure is different because communication mechanism between machines limits community structure correctness. Using the WIR community fairness evaluation metric in Section II-C, HBGR-METIS and HBGR-Vite outperform the original METIS and Vite in fairness (lower WIR values under various thresholds).

C. HBGR Algorithm

The main idea behind HBGR is to produce a new graph with weighted edges, where the magnitude and sign of the weights depend on the attributes of the nodes connected by each edge. Edges connecting nodes of different protected attributes are upweighted, indicating a stronger desire to keep these nodes on the same machine, while edges connecting nodes of the same protected attributes have lower weights. Pseudocode for HBGR is provided in Algorithm 1.

Algorithm 1: Homophily-Based Graph Reweighting Algorithm, HBGR

```
Input  : Graph G, a negative weight probability $p$
Output: Reweighted Graph G′
1 for each edge e in G.edges do
2     // Get two node attributes connected by the edge
      node1, node2 ← e.getNodes()
      attribute1 ← node1.getAttribute()
      attribute2 ← node2.getAttribute()
      // If the attributes are different, reweight the edge
      with a higher weight
      if attribute1 != attribute2 then
          $e.weight \leftarrow 2 \times \frac{|A_1|}{|N|} \times (1 - \frac{|A_1|}{|N|}) - \frac{|E_{A_1-A_2}|}{|E|}$
      else
          // If the attribute is the same, reweight the edge
          with a lower weight
          $e.weight \leftarrow (\frac{|A_1|}{|N|})^2 or (\frac{|A_2|}{|N|})^2 - \frac{|E_{A_1-A_2}|}{|E|}$
          if $random[0,1] > p$ then
              $e.weight \leftarrow -e.weight$
          else
              $e.weight \leftarrow e.weight$
      $G\prime.addedge(e.weight)$
3 return ($G\prime$)
```

1) **Edge Traversal:** HBGR traverses each edge in the graph to assign it a new weight. This process depends only on the attributes of the connected nodes, with no interdependence between the computations assigned to each edge. Accordingly, a parallel graph traversal algorithm can be used to improve efficiency. For example, using a single NVIDIA GPU to perform parallel BFS (Breadth-First Search) can traverse over 3.3 billion edges in one second [18].
2) **Cross-group Edge Reweighting:** For edges connecting nodes from different protected groups, HBGR assigns a greater weight: $2 \times \frac{|A_1|}{|N|} \times (1 - \frac{|A_1|}{|N|}) - \frac{|E_{A_1-A_2}|}{|E|}$. This

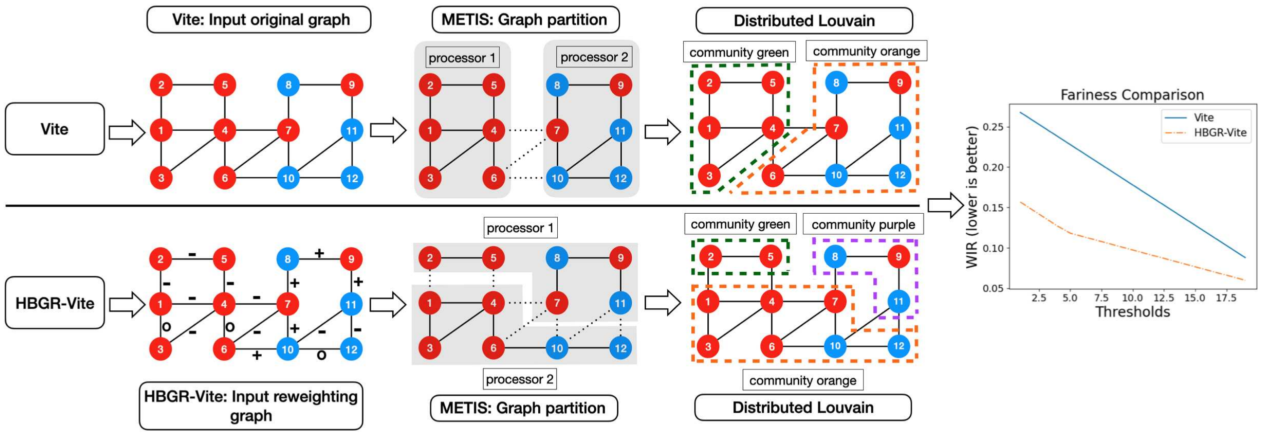

Fig. 1. A toy graph example demonstrating how HBGR protects edges connecting nodes with different attributes, and how the graph processed by HBGR-METIS graph partitioning results in fairer community detection outcome as compared to the METIS graph partition: Blue nodes are more likely to be contained in all communities, and HBGR-METIS Vite achieves better performance in terms of WIR.

reflects the homophily of the graph: the difference between the actual number of edges connecting two group nodes and their expected number of edges. The greater this difference, the more meaningful the connection, and so the higher the weight HBGR assigns to that edge. For weighted graphs, HBGR multiplies the original weight by $e.weight$ to apply various protection to different edges.

3) **Same-group Edge Reweighting:** For edges connecting same group nodes, HBGR assigns weight: $(\frac{|A_1|}{|N|})^2 or (\frac{|A_2|}{|N|})^2 - \frac{|E_{A_1-A_2}|}{|E|}$. This is the difference between the observed number of edges linking nodes within the same group and their expected number of edges. The greater the homophily value, meaning the more tightly connection between the vertices with the same protected group, the smaller the weight.
4) **Negative Weight Edge Selection:** To further encourage nodes from the same protected group to appear on different machines, HBGR randomly assigns some inter-group edges to have a negative weight. In our experiments, this was done with a probability of 0.5, which yielded good results across datasets.
5) **Output:** Output a copy $G\prime$ of graph G, where each edge is assigned its new weight.

Overall, HBGR generates a weighted graph based on node attributes distribution and graph homophily. This weighted graph can be applied to many distributed community discovery algorithms. Experimental results are provided in Section IV.

IV. Experiments

This section describes the dataset, experiment setup, evaluation methodologies, and experimental findings. Experiments demonstrate that HBGR improves fairness without significantly compromising accuracy or efficiency on selected datasets.

A. Datasets

We use the FaceBook100,[1] PokeC,[2] Twitch Gamers[3] and GitHub Social Network[4] for experimental evaluation. When there are missing attributes in datasets, we use networkx-nodeclassification[5] to predict the missing node traits. networkx-nodeclassfication is considered one of the best models for attribute prediction with regard to the prediction accuracy [19]. Detailed information of the dataset is in the table I.

	#nodes	#edges	Prot. Attr.
FaceBook100 Brown	8,600	384,526	Year
FaceBook100 Pennsylvania	41,554	1,362,229	Year
FaceBook100 Brandeis	3,898	137,567	Year
FaceBook100 Cal	11,247	35,1358	Year
FaceBook100 MIT	6,440	251,252	Year
FaceBook100 Northeastern	13,882	38,1934	Year
FaceBook100 Rice	4,087	184,828	Year
FaceBook100 Emory	7,460	330,014	Year
FaceBook100 Rutgers	24,580	784,602	Year
FaceBook100 Princeton	6596	293,320	Year
PokeC	1,632,803	36,022,564	gender
Twitch Gamers	168,144	6,797,557	maturity
GitHub Social Network	37,700	289,003	research areas

TABLE I
Dataset Statistics.

B. System Setup

We use Intel(R) Xeon(R) CPU E5-2690 v3 (2.60GHz, 30 MB Smart Cache) machines running Ubuntu Linux 22.04.2 ARM64 for our evaluation. GCC version 11.4.0 (Ubuntu 11.4.0-1ubuntu1 22.04) is used for the compilation. Our MPI implementation is based on OpenMPI version 4.1.2. In all distributed trials, 16 processors computed the Pokec dataset

[1]https://archive.org/details/oxford-2005-facebook-matrix
[2]https://snap.stanford.edu/data/soc-Pokec.html
[3]https://snap.stanford.edu/data/twitch_gamers.html
[4]https://snap.stanford.edu/data/github-social.html
[5]https://networkx.org/documentation/stable/reference/algorithms/index.html

and 8 tested the other datasets. All shown experimental results are the average of three tests.

C. Evaluation Metrics

In this section, we describe the accuracy, fairness, and efficiency metrics used to evaluate community detection algorithms.

1) Accuracy Evaluation: We use modularity to evaluate community accuracy. Modularity is the difference between the ratio of the total number of edges within communities and the total number of edges in the network, and an expected value is the size of the ratio when the network is set to a random network formed by the same community [13]. Positive and larger modularity values indicate better community structure.

2) Fairness Evaluation: As discussed in Section II-C, lower WIR values indicate better fairness performance. In our trade-off analysis across accuracy, fairness, and efficiency, we use a new measurement standard: WIR_{sum}, to evaluate the overall fairness performance of an algorithm across thresholds. WIR_{sum} is the integral of the WIR curve within the region of thresholds ranging from 1% to 30%, estimated by calculating WIR at 1% intervals.

3) Efficiency Evaluation: To evaluate efficiency, we use the concept of speed-up to quantify the acceleration of distributed systems compared to sequential algorithms. The speed-up achieved by a distributed method is determined by comparing the time it takes for the best sequential approach to solve a task and it can be calculated by: $speedup = \frac{T_S}{T_D}$, where T_S and T_D represent the computation time by using sequential algorithm and distributed framework respectively. The larger this value, the greater the efficiency improvement.

D. Results

Fig. 2 shows examples of the fairness performance of METIS-based Louvain (Vite) and HBGR-based Louvain on four FaceBook100 datasets (Brown, Northeastern, Brandeis, and Penn). As seen in the figure, HBGR-METIS-based Louvain has a lower WIR value across thresholds than METIS-distributed Louvain, indicating a more fairn community structure.

Table II contains full results of the accuracy, fairness, and efficiency of the considered algorithms. We test performance with these Louvain and InfoMap's corresponding sequential algorithms and the original distributed algorithms. Section IV-C introduces relevant metrics: higher modularity indicates better accuracy, lower WIR shows better fairness, and higher speedup indicates superior efficiency.

The data in Table II shows that using the HBGR-METIS algorithm in distributed community detection frameworks results in generally superior fairness performance, while maintaining efficiency levels and with only a small reduction in accuracy. In Section V, we analyze the trade-off between correctness, fairness, and efficiency in this setting.

V. Discussion

As can be seen in Table II (percentage improvement or losing), there is not a significant fairness-efficiency tradeoff:

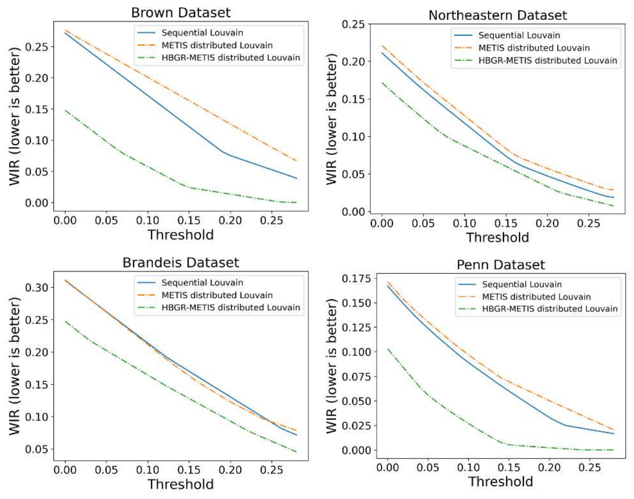

Fig. 2. Fairness comparison between sequential Louvain, METIS distributed Louvain, and HBGR-METIS distributed Louvain on multiple FaceBook100 data sets (using attribute threshold matriculation year of 2007), based on WIR. A community structure is fairer if it has a lower WIR.

indeed, in many cases, efficiency improves when using HBGR-METIS vs. standard METIS. Over half of the results showed that HBGR could produce a small improvement in efficiency (ranging from 1% to 31%), and in the remaining half of experiments where there was an efficiency loss, the loss typically did not exceed 10% (with the only exceptions being the Cal dataset: HBGR-distributed Louvain had a loss of 36.24%, and HBGR-RelaxMap had a loss of 25.79%).

There is a tradeoff between fairness and accuracy– as is common in many fair algorithms– but we see that HBGR achieves more than a 30% (and in many cases more than a 50%) improvement in fairness with, typically, no more than a 15% loss in accuracy. Additionally, we discover that when HBGR loses more accuracy compared to the standard method in a specific distributed clustering framework, fairness can be greatly improved, as shown in datasets Facebook100 Penn ($+68.96\%$), Facebook100 Brown ($+71.61\%$), and Facebook100 Princeton ($+68.53\%$) for distributed Louvain. When the accuracy performance of HBGR is similar to that of the standard method, the improvement in fairness is smaller, but still significant, as seen on the Facebook100 Brandeis ($+23.61\%$), Facebook100 Rice($+23.64\%$), and PokeC ($+17.00\%$) datasets with distributed Louvain.

VI. Conclusion and Future Work

In this paper, we proposed the Homophily-Based Graph Reweighting (HBGR) algorithm that can be applied to different distributed community detection frameworks. Evaluation on thirteen datasets demonstrated better fairness performance as compared to the standard distributed clustering approaches. To the best of our knowledge, this is the first graph algorithm aimed at reducing fairness issues of distributed community detection systems. Through a trade-off analysis of accuracy, fairness, and efficiency, we demonstrated that HBGR achieves better fairness performance while maintaining

		Accuracy comparison based on modularity			Fairness comparison based on WIR			Efficiency comparison based on speed up	
		Sequential	Standard Distributed	HBGR Distributed	Sequential	Standard Distributed	HBGR Distributed	Standard Distributed	HBGR Distributed
Louvain	Brown	0.3522	**0.3937**	0.3713 (-5.11%)	4.0445	4.9555	**1.4071** (+71.61%)	1.58x	**1.65x** (+4.43%)
	Penn	0.4243	**0.4438**	0.3846 (-13.34%)	2.1285	2.4033	**0.7459** (+68.96%)	**3.34x**	3.12x (-6.59%)
	Brandeis	0.3832	**0.4087**	0.4056 (-0.76%)	5.3131	5.2442	**4.0058** (+23.61%)	3.85x	**4.39x** (+14.03%)
	Cal	0.4311	0.3070	**0.3218** (+4.82%)	3.1030	2.3876	**1.9582** (+17.98%)	**4.69x**	2.99x (-36.24%)
	MIT	0.3183	0.3498	**0.3519** (+0.60%)	2.4858	3.0519	**2.7306** (+10.53%)	**3.24x**	3.18x (-1.86%)
	Northeastern	0.4358	**0.4347**	0.3850 (-11.43%)	2.7712	3.0612	**2.0940** (+31.59%)	2.77x	**3.15x** (+13.72%)
	Rice	0.42201	**0.3944**	0.3741 (-5.14%)	0.6968	1.1764	**0.8982** (+23.64%)	3.64x	**4.18x** (+14.83%)
	Emory	0.4733	**0.4802**	0.3824 (-20.37%)	3.6648	4.2050	**1.1276** (+73.18%)	1.91x	**1.95x** (+2.09%)
	Rutgers	0.4716	**0.4389**	0.4298 (-2.07%)	1.073	2.6641	**1.7012** (+36.14%)	4.38x	**4.48x** (+2.28%)
	Princeton	0.4535	**0.4529**	0.3529 (-22.08%)	8.3336	8.7014	**2.7381** (+68.53%)	4.04x	**4.10x** (+1.49%)
	PokeC	0.7134	**0.6812**	0.6599 (-3.13%)	4.6587	4.9455	**4.1048** (+17.00%)	**6.15x**	5.57x (-9.43%)
	Twitch Gamers	0.4138	**0.4253**	0.3831 (-9.92%)	3.1547	3.8438	**3.2217** (+16.18%)	4.29x	**4.36x** (+1.63%)
	GitHub	0.4539	**0.4653**	0.4247 (-8.72%)	2.5714	3.9555	**2.4071** (+39.15%)	3.74x	**4.27x** (+14.17%)
InfoMap	Brown	0.3347	**0.3341**	0.2142 (-35.89%)	4.7724	4.9018	**2.3012** (+53.05%)	1.66x	**1.85x** (+11.44%)
	Penn	0.3895	**0.3479**	0.3002 (-13.73%)	2.7611	3.1960	**2.2393** (+29.93%)	1.75x	**1.77x** (+1.14%)
	Brandeis	0.1857	**0.1953**	0.1423 (-27.14%)	2.2779	2.3462	**1.6902** (+27.96%)	1.25x	**1.39x** (+10.07%)
	Cal	0.3373	**0.3365**	0.2768 (-17.74%)	2.7218	3.5334	**2.9264** (+17.18%)	**2.21x**	1.64x (-25.79%)
	MIT	0.2501	0.2259	**0.2577** (+14.07%)	2.3698	2.6243	**1.9429** (+25.97%)	**1.31x**	1.18x (-9.92%)
	Northeastern	0.3636	**0.3258**	0.2574 (-20.99%)	3.3736	3.6633	**2.5207** (+31.19%)	**1.49x**	1.37x (-8.05%)
	Rice	0.3819	0.1040	**0.3732** (+258%)	1.5923	2.6471	**1.1095** (+58.08%)	2.02x	**2.51x** (+24.26%)
	Emory	0.3646	**0.3554**	0.3384 (-4.78%)	2.5134	3.2670	**2.5820** (+20.97%)	1.13x	**1.44x** (+21.53%)
	Rutgers	0.3766	**0.3265**	0.2835 (-13.17%)	2.9806	3.9005	**2.4911** (+36.13%)	1.76x	**2.30x** (+30.68%)
	Princeton	0.3763	**0.2764**	0.2589 (-6.33%)	5.0165	4.9549	**2.5322** (+48.90%)	3.05x	**3.15x** (+3.28%)
	PokeC	0.6076	**0.5302**	0.4876 (-8.03%)	4.1732	4.1169	**3.3545** (+18.53%)	4.37x	**4.98x** (+13.96%)
	Twitch Gamers	0.2471	**0.2219**	0.1973 (-11.09%)	2.8441	3.8185	**2.8874** (+24.38%)	**1.80x**	1.71x (-5.00%)
	GitHub	0.2918	**0.1999**	0.1753 (-12.31%)	2.8038	3.0397	**2.1865** (+28.07%)	3.15x	**2.87x** (-8.89%)

TABLE II
COMPLETE EVALUATION RESULTS OF TWO ALGORITHMS (LOUVAIN AND INFOMAP) WITH THREE VERSIONS: SEQUENTIAL ALGORITHM, STANDARD DISTRIBUTED FRAMEWORKS, AND HBGR-BASED DISTRIBUTED APPROACH. IN THE ACCURACY COMPARISON, A LARGER MODULARITY VALUE INDICATES A BETTER COMMUNITY STRUCTURE. IN THE FAIRNESS COMPARISON, A SMALLER WIR VALUE REPRESENTS BETTER FAIRNESS PERFORMANCE. IN THE EFFICIENCY COMPARISON, A LARGER SPEED-UP VALUE SUGGESTS A BETTER RUNNING TIME ENHANCEMENT. THE BOLDED DATA IN THE FIGURE INDICATES WHICH OF THE TWO–THE STANDARD DISTRIBUTED ALGORITHM OR THE HBGR DISTRIBUTED ALGORITHM– IS SUPERIOR IN TERMS OF ACCURACY, FAIRNESS, OR EFFICIENCY. THE PERCENTAGES IN PARENTHESES REFLECT THE DEGREE OF IMPROVEMENT OR DECLINE OF THE HBGR DISTRIBUTED ALGORITHM RELATIVE TO THE STANDARD DISTRIBUTED ALGORITHM IN THESE THREE METRICS.

the current operational efficiency and not sacrificing much accuracy. We leave the application of HBGR to other distributed graph algorithms to achieve more fair results for future work.

REFERENCES

[1] Aggarwal, C.C., 2011. An introduction to social network data analytics (pp. 1-15). Springer US.

[2] Fortunato, S., 2010. Community detection in graphs. Physics reports, 486(3-5), pp.75-174.

[3] McCune, R.R., Weninger, T. and Madey, G., 2015. Thinking like a vertex: A survey of vertex-centric frameworks for large-scale distributed graph processing. ACM Computing Surveys (CSUR), 48(2), pp.1-39.

[4] Ghosh, S., Halappanavar, M., Tumeo, A., Kalyanaraman, A., Lu, H., Chavarria-Miranda, D., Khan, A. and Gebremedhin, A., 2018, May. Distributed louvain algorithm for graph community detection. IEEE IPDPS 2018.

[5] Zhang, H., Jayaweera, M., Ren, B., Wang, Y., Soundarajan, S. (2023, December). Unfairness in distributed graph frameworks. In 2023 IEEE International Conference on Data Mining (ICDM) (pp. 1529-1534). IEEE.

[6] Saxena, A., Fletcher, G. and Pechenizkiy, M., 2024. Fairsna: Algorithmic fairness in social network analysis. ACM Computing Surveys, 56(8), pp.1-45.

[7] Kang, J., He, J., Maciejewski, R. and Tong, H., 2020, August. Inform: Individual fairness on graph mining. In Proceedings 26th ACM SIGKDD.

[8] Manolis, K. and Pitoura, E., 2023, November. Modularity-Based Fairness in Community Detection. In Proceedings of ASONAM 2023.

[9] Bae, S.H., Halperin, D., West, J.D., Rosvall, M. and Howe, B., 2017. Scalable and efficient flow-based community detection for large-scale graph analysis. ACM Transactions on Knowledge Discovery from Data (TKDD), 11(3), pp.1-30.

[10] Stanton, I. and Kliot, G., 2012, August. Streaming graph partitioning for large distributed graphs. In Proceedings of the 18th ACM SIGKDD.

[11] Karypis, G. and Kumar, V., 1997. METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices.

[12] Chierichetti, Flavio, et al. 2017. Fair clustering through fairlets. Advances in neural information processing systems, 30.

[13] Newman, M.E., 2006. Modularity and community structure in networks. Proceedings of the national academy of sciences, 103(23), pp.8577-8582.

[14] McPherson, M., Smith-Lovin, L. and Cook, J.M., 2001. Birds of a feather: Homophily in social networks. Annual review of sociology, 27(1), pp.415-444.

[15] Kleinberg, J., Ludwig, J., Mullainathan, S. and Rambachan, A., 2018, May. Algorithmic fairness. In Aea papers and proceedings (Vol. 108, pp. 22-27). 2014 Broadway, Suite 305, Nashville, TN 37203: American Economic Association.

[16] Blondel, V.D., Guillaume, J.L., Lambiotte, R. and Lefebvre, E., 2008. Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment, 2008(10), p.P10008.

[17] Rosvall, M. and Bergstrom, C.T., 2008. Maps of random walks on complex networks reveal community structure. Proceedings of the national academy of sciences, 105(4), pp.1118-1123.

[18] Merrill, D., Garland, M. and Grimshaw, A., 2012. Scalable GPU graph traversal. ACM Sigplan Notices, 47(8), pp.117-128.

[19] Zhu, X., Ghahramani, Z. and Lafferty, J.D., 2003. Semi-supervised learning using gaussian fields and harmonic functions. In Proceedings of ICML-03.