On Generalized Degree Fairness in Graph Neural Networks (Technical Appendices)

Zemin Liu[1*], **Trung-Kien Nguyen**[2], **Yuan Fang**[2]

[1] National University of Singapore, Singapore
[2] Singapore Management University, Singapore
zeminliu@nus.edu.sg, {tknguyen, yfang}@smu.edu.sg

A Algorithm and Complexity

In this section, we first provide an outline of the model training, then analyze its computational complexity.

Algorithm. We outline the model training for DegFairGNN in Algorithm 1. In line 1, we initialize the parameters set Θ. In line 2, the training set $\mathcal{V}^{\text{tr}}$ is divided into S_0 and S_1 w.r.t. the predefined threshold K. In lines 4–10, we calculate the layer-wise node representations for the training set $\mathcal{V}^{\text{tr}}$. In particular, for a target node v in layer l, the context embedding $\mathbf{c}_v^l$ is calculated in line 6, and the scaling and shifting vectors γ_v^l and β_v^l are calculated in line 7. Furthermore, the debiasing contexts for complementing (*i.e.*, $\mathcal{D}(v;\theta_0^l)$) and distilling (*i.e.*, $\mathcal{D}(v;\theta_1^l)$) are achieved in lines 8 and 9, respectively. Thereafter, we generate the node representation $\mathbf{h}_v^l$ based on the debiasing neighbor aggregation in line 10. In lines 11–14, we compute the classification loss (line 11), fairness loss (line 12), constraints on debiasing contexts (line 13), and constraints on scaling and shifting vectors (line 14), respectively. Based on them, we formalize the overall objective $\mathcal{L}$ in line 15. We optimize Θ by minimizing $\mathcal{L}$ in line 16.

Complexity analysis. The debiasing neighborhood aggregation increases the computational cost. Taking GCN as base model, with everything else being the same, we compare the complexity of 1-layer neighborhood aggregation for one node in GCN and DegFairGCN. Given a node v with degree d, the complexity of GCN is $O(d)$ for neighborhood aggregation. On the other hand, the debiasing neighborhood aggregation of DegFairGCN contains the following steps. (1) The calculation of context embedding $\mathbf{c}_v^l$: we use an operator of mean-pooling to aggregate the layer-l contents in node v's r-hop local context $\mathcal{C}_r(v)$ with $r=1$, thus involving the complexity of $O(d)$. (2) The calculation of scaling (*i.e.*, γ_v^l) and shifting (*i.e.*, β_v^l) vectors: we utilize degree encoding $\delta^l(v)$ to calculate the scaling and shifting vectors, with complexity $O(1)$ for both of them. (3) The calculation of debiasing contexts, including both $\mathcal{D}(v;\theta_0^l)$ and $\mathcal{D}(v;\theta_1^l)$: they are computed based on the achieved context embedding and scaling and shifting vectors, thus resulting in complexity of $O(1)$. (4) The calculation of debiasing neighborhood aggregation $\mathbf{h}_v^l$: with referring to the neighbors once again for neighborhood aggregation (*i.e.*, operation $\textsc{Aggr}(\cdot)$), this step involves additional complexity of $O(d)$. Overall, in the calculation of debiasing neighborhood aggregation, the complexity of DegFairGCN is $O(2d+2)$, which has the same computational scale with its base model GCN, only differing by a constant factor. Note that, this comparison is also appropriate for other GNNs and the corresponding DegFairGNNs.

Algorithm 1: MODEL TRAINING FOR DEGFAIRGNN

```
Require: Graph $\mathcal{G}=\{\mathcal{V},\mathcal{E},\mathbf{X}\}$, training set $\mathcal{V}^{\text{tr}}$, threshold hyperparameter $K$, and hyperparameters $\{\epsilon,\mu,\lambda\}$.
Ensure: Model parameters $\Theta$.
 1: $\Theta \leftarrow$ parameters initialization;
 2: Split training set $\mathcal{V}^{\text{tr}}$ into $S_0$ and $S_1$ w.r.t. $K$;
 3: while not converged do
 4:     for $l \in \{1,2,\ldots,\ell\}$ do
 5:         for $v \in \mathcal{V}^{\text{tr}}$ do
 6:             $\mathbf{c}_v^l \leftarrow \textsc{Pool}(\{\mathbf{h}_u^{l-1} \mid u \in \mathcal{C}_r(v)\})$;
 7:             $\gamma_v^l \leftarrow \phi_\gamma(\delta^l(v);\theta_\gamma^l), \quad \beta_v^l \leftarrow \phi_\beta(\delta^l(v);\theta_\beta^l)$;
 8:             $\mathcal{D}(v;\theta_0^l) \leftarrow (\gamma_v^l+\mathbf{1}) \odot f(\mathbf{c}_v^l;\theta_{c,0}^l) + \beta_v^l$;
 9:             $\mathcal{D}(v;\theta_1^l) \leftarrow (\gamma_v^l+\mathbf{1}) \odot f(\mathbf{c}_v^l;\theta_{c,1}^l) + \beta_v^l$;
10:             $\mathbf{h}_v^l \leftarrow$ debiasing neighbor aggregation by Eq. (9);
11:     $\mathcal{L}_1 \leftarrow -\sum_{v\in\mathcal{V}^{\text{tr}}}\sum_{y=1}^{|\mathcal{Y}|}[\mathbf{y}_v]_y \ln[\mathbf{h}_v^\ell]_y$;
12:     $\mathcal{L}_2 \leftarrow \left\| \frac{1}{|S_0^{\text{tr}}|}\sum_{v\in S_0^{\text{tr}}}\mathbf{h}_v^\ell - \frac{1}{|S_1^{\text{tr}}|}\sum_{v\in S_1^{\text{tr}}}\mathbf{h}_v^\ell \right\|_2^2$;
13:     $\mathcal{L}_3 \leftarrow \sum_{l=1}^{\ell}\left(\sum_{v\in\mathcal{S}_0^{\text{tr}}}\|\mathcal{D}(v;\theta_1^l)\|_2^2 + \sum_{v\in\mathcal{S}_1^{\text{tr}}}\|\mathcal{D}(v;\theta_0^l)\|_2^2\right)$;
14:     $\mathcal{L}_4 \leftarrow \sum_{l=1}^{\ell}\sum_{v\in\mathcal{V}_{\text{tr}}}(\|\gamma_v^l\|_2^2 + \|\beta_v^l\|_2^2)$;
15:     $\mathcal{L} \leftarrow \mathcal{L}_1 + \mu\mathcal{L}_2 + \lambda\cdot(\mathcal{L}_3+\mathcal{L}_4+\Omega(\Theta))$;
16:     Update $\Theta$ by minimizing $\mathcal{L}$;
17: return $\Theta$.
```

B Details of Datasets

We use two Wikipedia networks, namely *Chameleon* and *Squirrel* (Pei et al. 2020), in which each node represents a Wikipedia page, and each edge denotes a reference between two pages in either direction. Node features are derived from informative keywords on each page. Each node is associated with a number denoting the monthly traffic of the page.

[*]Part of the work was done as a research scientist at Singapore Management University.

We split the nodes into five categories w.r.t. their traffic volume for node classification. We also use a citation network *EMNLP* (Ma et al. 2021), in which each node denotes a paper published in the EMNLP conference, and each edge denotes two papers that have been co-cited by at least two EMNLP papers. Node features are derived from the principal components of keywords in the paper title/abstract and the year of publication. Each node is associated with a number denoting the citation from outside of EMNLP. We split the nodes into two categories w.r.t. their outside EMNLP citation count for node classification.

C Details of Base GNN Models

We give detailed descriptions and settings for the base GNN models, including GCN (Kipf and Welling 2017), GAT (Veličković et al. 2018) and GraphSAGE (Hamilton, Ying, and Leskovec 2017).

Descriptions. We describe the three base GNN models below.

- GCN (Kipf and Welling 2017): GCN relies on the convolutional operation of neighborhood aggregation to recursively incorporate information from the neighbors of a target node. In particular, it resorts to mean-pooling to aggregate information from neighbors to produce the node representations.
- GAT (Veličković et al. 2018): GAT utilizes a self-attention-based neighborhood aggregation in each layer, thus the neighbors can be assigned particular weights w.r.t. their content.
- GraphSAGE (Hamilton, Ying, and Leskovec 2017): GraphSAGE relies on a similar yet different way with GCN for neighborhood aggregation, by paying more attention to the information from the target node itself.

D Details of Baselines

Descriptions. We describe the baselines as follows, which are either degree-specific models or fairness-aware models.

The first group of baselines are *degree-specific models.*

- DSGCN (Tang et al. 2020): To address accuracy bias among nodes with different degrees, DSGCN proposes to employ different models for nodes with different degrees, thus they can improve the performance for all the nodes.
- Residual2Vec (Kojaku et al. 2021): Residual2Vec also addresses a similar problem with DSGCN, which tries to modulate the random walk w.r.t. the node degrees thus they can achieve balanced probability for node sampling for better accuracy on all the nodes.

The second group of baselines are *fairness-aware models.*

- FairWalk (Rahman et al. 2019): FairWalk follows the same paradigm of DeepWalk (Perozzi, Al-Rfou, and Skiena 2014), including paths sampling and node representation learning (*e.g.*, by word2vec (Mikolov et al. 2013)). In particular, they conduct a sensitive attribute guided walk for path sampling. In other words, when sampling from the neighbors, they first randomly select a sensitive attribute, then randomly select a node from the chosen attribute.
- CFC (Bose and Hamilton 2019): In virtue of the well-designed filters over the node representations, CFC can filter out the sensitive attributes from the node representations for debiasing. It further employs a discriminator to adversarially facilitate the identification of sensitive attributes.
- FairGNN (Dai and Wang 2021): Similar to CFC, FairGNN also applies the adversarial training principle for the promotion of distinguishing sensitive attributes from the node representations.
- FairAdj (Li et al. 2021): For dyadic fairness which is a distinct problem from ours, FairAdj tries to learn a fair adjacency matrix with proper graph structural constraints to debias the sensitive attributes for fair link prediction, and maintain the prediction accuracy at the same time.
- FairVGNN (Wang et al. 2022): FairVGNN tries to address the issue of sensitive attribute leakage. To achieve this, they resort to a generative adversarial debiasing module to obtain fair node representations, and an adaptive weight clamping module to clamp the weights of the encoder based on learned fair feature views.

E Hyper-Parameter Settings

General settings. For all the GNN-based models, we apply a two-layer architecture (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Veličković et al. 2018).

Settings of base GNNs. Based on the proposed setup by the original papers, we further tune the parameters in order to achieve their optimal performance, including accuracy and fairness. For GCN (Kipf and Welling 2017), we set the dropout rate as 0.5, the coefficient of regularization as 0.0001, and the learning rate as 0.01. In particular, we set the hidden dimension of the first layer as 32 for Chameleon, 64 for Squirrel and 16 for EMNLP. For GAT (Veličković et al. 2018), we utilize a self-attention mechanism with two heads, and set the dropout rate as 0.5, the coefficient of regularization as 0.0001, the learning rate as 0.01, and the hidden dimension of the first layer as 16 for all the datasets. For GraphSAGE (Hamilton, Ying, and Leskovec 2017), we use the mean-pooling as the aggregator, and set the dropout rate as 0.5, the coefficient of regularization as 0.0001, and the learning rate as 0.01. In particular, we further set the hidden dimension of the first layer as 32 for Chameleon and Squirrel and 16 for EMNLP.

Settings of baselines. We tune the parameters for baselines based on the proposed settings from the original papers to achieve their optimal performance, including accuracy and fairness. For DSGCN (Tang et al. 2020), we implement the model ourselves based on the details illustrated in the paper, and set the hidden dimension of the first layer as 32, and the threshold d_{max} to prevent the long-tail issue of the degrees as 20. For Residual2Vec (Kojaku et al. 2021), we choose the version of matrix factorization for model optimization, and set the hidden dimension of embeddings as 32, and the length of context window as 10. For FairWalk

(Rahman et al. 2019), we sample 20 paths starting from each node with length 80, and set embedding dimension as 128 on Chameleon and Squirrel while 64 on EMNLP w.r.t. their performance. For CFC (Bose and Hamilton 2019), to achieve its optimal performance, we set the training steps of the discriminator as 2 with 0.1 as its weight, and the hidden dimension of the first layer as 32 on all the three datasets. For FairGNN (Dai and Wang 2021), we set $\alpha = 4$ (controlling the influence of the adversary to the GNN classifier) and β (controlling the contribution of the covariance constraint to ensure fairness) as 1 for Chameleon, 0.01 for Squirrel, 10 for EMNLP, and the hidden dimension of the first layer as 32 on the three datasets. For FairAdj (Li et al. 2021), we replace the reconstruction loss with node classification loss for fair comparison, and set the number of epochs for utility optimization T_1 as 100. In addition, on datasets Chameleon and Squirrel, we set the number of epochs for fairness optimization T_2 as 10, and η as 0.2; on dataset EMNLP, we set T_2 as 5 and η as 0.1. For FairVGNN (Wang et al. 2022), we set the number of views K as 5, clamp weight ϵ as 1, and mask density weight α as 1.

Settings of DegFairGNN. We build DegFairGNNs based on the three base GNN models including GCN (Kipf and Welling 2017), GAT (Veličković et al. 2018) and GraphSAGE (Hamilton, Ying, and Leskovec 2017), thus forming the extended versions including DegFairGCN, DegFairGAT and DegFairSAGE, respectively. In particular, we employ a group of unified main parameters for all these three extensions, by setting $\lambda = 0.0001$, learning rate as 0.01 and dropout rate as 0.5. We also uniformly tune the other parameters for them on each dataset: on Chameleon, we set the hidden dimension of the first layer as 32, $\epsilon = 1$ and $\mu = 0.001$; on Squirrel, the hidden dimension as 32, $\epsilon = 1$ and $\mu = 0.0001$; and on EMNLP, the hidden dimension as 16, $\epsilon = 0.001$ and $\mu = 0.01$. As an exception, we set the hidden dimension of the first layer as 16 for DegFairGAT on all the three datasets, and deploy the self-attention mechanism with three heads.

F Environment

We implemented the proposed DegFairGNN using Pytorch 1.6 in Python 3.6.5. All experiments were conducted on a Linux workstation with a 6-core 3.6GHz CPU, 128GB DDR4 memory and two RTX 2080Ti GPUs.

G Further Experiments and Analysis

G.1 Additional Fairness Settings Using Different Base GNNs

We report the comparison of extensions with different base GNN models (*i.e.*, GAT and GraphSAGE) for the setting of $r = 2$ with 20% Top/Bottom and $r = 1$ with 30% Top/Bottom in Tables I and II, respectively. Note that, accuracy evaluation is applied on the whole test set regardless of the groups, so the accuracies reported in these two tables are identical to those in Table 5 in the main paper. We can observe that, across different fairness evaluation settings, our proposed DegFairGAT and DegFairSAGE can generally achieve superior performance than their corresponding base GNN models in terms of degree fairness, while achieving comparable and even better accuracy. This phenomenon further demonstrates the generalizability of our proposed DegFairGNN to different backbones in various fairness evaluation settings.

Table I: With other base GNNs ($r = 2$, 20% Top/Bottom).

		GAT	DegFairGAT	GraphSAGE	DegFairSAGE
Chamel.	Acc. $\uparrow$	63.15 ± 0.40	69.64 ± 0.44	53.15 ± 0.56	60.95 ± 0.84
	$\Delta_{\text{DSP}} \downarrow$	8.74 ± 1.03	$\mathbf{5.58} \pm 1.45$	8.92 ± 0.67	$\mathbf{7.78} \pm 1.36$
	$\Delta_{\text{DEO}} \downarrow$	28.58 ± 1.52	$\mathbf{20.73} \pm 2.27$	29.50 ± 2.37	$\mathbf{21.83} \pm 1.65$
Squirrel	Acc. $\uparrow$	41.44 ± 0.21	45.55 ± 1.44	34.39 ± 0.62	34.63 ± 1.31
	$\Delta_{\text{DSP}} \downarrow$	15.20 ± 1.16	$\mathbf{10.89} \pm 1.67$	6.92 ± 0.63	$\mathbf{3.89} \pm 0.53$
	$\Delta_{\text{DEO}} \downarrow$	26.74 ± 1.18	$\mathbf{20.76} \pm 3.24$	17.12 ± 3.28	$\mathbf{14.25} \pm 1.58$
EMNLP	Acc. $\uparrow$	70.42 ± 0.77	81.57 ± 1.14	83.96 ± 0.31	83.57 ± 0.44
	$\Delta_{\text{DSP}} \downarrow$	24.04 ± 2.90	$\mathbf{13.26} \pm 7.11$	54.06 ± 1.15	$\mathbf{25.96} \pm 3.55$
	$\Delta_{\text{DEO}} \downarrow$	$\mathbf{7.89} \pm 1.27$	11.37 ± 6.09	49.31 ± 1.06	$\mathbf{21.76} \pm 3.15$

Table II: With other base GNNs ($r = 1$, 30% Top/Bottom).

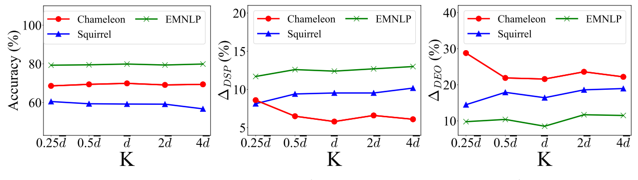
		GAT	DegFairGAT	GraphSAGE	DegFairSAGE
Chamel.	Acc. $\uparrow$	63.15 ± 0.40	69.64 ± 0.44	53.15 ± 0.56	60.95 ± 0.84
	$\Delta_{\text{DSP}} \downarrow$	5.09 ± 0.55	$\mathbf{4.29} \pm 1.16$	9.27 ± 0.78	$\mathbf{8.11} \pm 0.76$
	$\Delta_{\text{DEO}} \downarrow$	17.68 ± 0.97	$\mathbf{15.62} \pm 1.95$	24.31 ± 2.55	$\mathbf{17.69} \pm 1.76$
Squirrel	Acc. $\uparrow$	41.44 ± 0.21	45.55 ± 1.44	34.39 ± 0.62	34.63 ± 1.31
	$\Delta_{\text{DSP}} \downarrow$	10.30 ± 0.82	$\mathbf{10.03} \pm 0.58$	4.54 ± 0.36	$\mathbf{3.74} \pm 0.37$
	$\Delta_{\text{DEO}} \downarrow$	20.25 ± 0.77	$\mathbf{17.77} \pm 3.48$	14.70 ± 1.11	$\mathbf{14.54} \pm 1.44$
EMNLP	Acc. $\uparrow$	70.42 ± 0.77	81.57 ± 1.14	83.96 ± 0.31	83.57 ± 0.44
	$\Delta_{\text{DSP}} \downarrow$	27.58 ± 2.27	$\mathbf{14.06} \pm 4.47$	50.55 ± 1.52	$\mathbf{28.77} \pm 2.88$
	$\Delta_{\text{DEO}} \downarrow$	$\mathbf{13.33} \pm 2.09$	14.24 ± 4.55	51.84 ± 1.46	$\mathbf{29.12} \pm 2.91$

G.2 Impact of Threshold K

Recall that we set the threshold K for the structural contrast as the mean node degree $\bar{d}$ during training. In Fig. I, we further investigate its impact on the model by varying K in $\{0.25\bar{d}, 0.5\bar{d}, \bar{d}, 2\bar{d}, 4\bar{d}\}$. Firstly, we observe that model accuracy remains stable across different values of K, showing that the choice of K has little impact on the accuracy since K is mainly employed to modulate degree fairness. Secondly, in fairness metrics Δ_{DSP} and Δ_{DEO}, DegFairGCN generally achieves optimal performance with $K = \bar{d}$, and smaller or larger values of K tend to impair the fairness. This demonstrates that the mean node degree is a good choice for structural contrast in training.

G.3 Scalability

In order to evaluate the scalability of DegFairGNN when applying to large graphs, we run DegFairGCN and baseline FairGNN on a large dataset Amazon (Liu, Nguyen, and Fang 2021) for comparison. In particular, we sample a set of graphs on Amazon with graph sizes (the number of nodes) 20k, 40k, 60k, 80k, 100k and 1M, and split the nodes into training, validation and testing set with proportion 6:2:2. We report the training (per epoch) and inference time (ms) in Table III. We observe that the time cost of DegFairGCN generally increases linearly as the graph sizes increase, showing its scalability on large graphs. Besides, DegFairGCN can generally cost less time than FairGNN across different graph sizes.



(a) Accuracy ↑ (b) Δ_{DSP} ↓ (c) Δ_{DEO} ↓

Figure I: Impact of threshold K.

Table III: Scalability comparison of FairGNN and DegFairGCN in terms of training (per epoch) and inference time (ms).

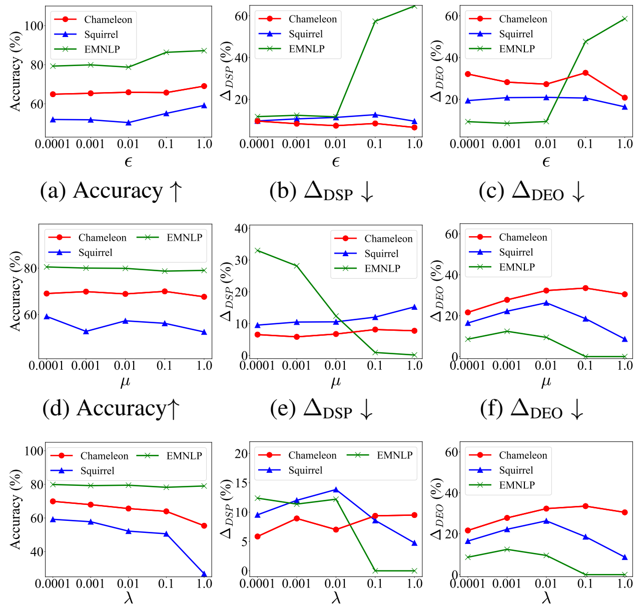
Graph sizes (# nodes)		20k	40k	60k	80k	100k	1M
Training time (ms)	FairGNN	11	42	90	OOM	OOM	OOM
	DegFairGCN	11	18	29	43	49	570
Inference time (ms)	FairGNN	9	38	80	OOM	OOM	OOM
	DegFairGCN	4	7	10	14	20	270

G.4 Parameter Sensitivity

In Fig. II, we further study the sensitivity of other hyperparameters including ϵ, μ and λ. We first show the impact of parameter ϵ in Figs. II (a)-(c), and have the following observations. (1) Higher ϵ (*e.g.*, [0.1,1.0]) might slightly boost the accuracy, showing that modulating the neighborhood aggregation can potentially enhance and denoise the node representations. (2) The range [0.001, 0.01] is generally robust for fairness while maintaining competitive accuracy. Next, Figs. II (d)-(f) show the influence of parameter μ. (1) The accuracy is generally stable with different values of μ. (2) When $\mu \in [0.1, 1.0]$, both Δ_{DSP} and Δ_{DEO} on EMNLP generally approach 0 since this extreme setting of μ drives the model to gradually predict the same label for all nodes on EMNLP due to the large weight of fairness loss. Lastly, we illustrate the impact of λ in Figs. II (g)-(i). For both Δ_{DSP} and Δ_{DEO}, larger λ (*e.g.*, 1.0) can generally promote the fairness metrics (except Δ_{DEO} on Chameleon), while it may impair the accuracy. A good trade-off is $\lambda \in [0.01, 0.1]$, which achieves generally robust performance in both accuracy and fairness. Lastly, we illustrate the impact of λ in Figs. II (g)-(i). For both Δ_{DSP} and Δ_{DEO}, smaller or larger λ's can generally promote the fairness metrics, while it may impair the accuracy when λ is larger. A good trade-off is $\lambda \in [0.0001, 0.001]$, which achieves generally robust performance in both accuracy and fairness.

H Additional Related Work

Graph representation learning. To overcome the high cost of feature engineering in traditional graph mining algorithms, graph embedding (Cai, Zheng, and Chang 2018; Perozzi, Al-Rfou, and Skiena 2014; Tang et al. 2015; Grover and Leskovec 2016) and graph neural networks (GNNs) (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Veličković et al. 2018; Xu et al. 2019) open up great opportunities for representation learning on graphs, which embed graph elements (*e.g.*, nodes, edges, subgraphs or even the whole graph) into low-dimensional vectors to preserve the graph structures.



(a) Accuracy ↑ (b) Δ_{DSP} ↓ (c) Δ_{DEO} ↓

(d) Accuracy↑ (e) Δ_{DSP} ↓ (f) Δ_{DEO} ↓

(g) Accuracy↑ (h) Δ_{DSP} ↓ (i) Δ_{DEO} ↓

Figure II: Influence of hyperparameters ϵ, μ and λ.

Other related studies. Fairness studies for recommendation systems usually require additional side-information (Beutel et al. 2019), or consider multi-sided fairness arising on the consumer side, provider side or both sides (Burke, Sonboli, and Ordonez-Gauger 2018; Mehrotra et al. 2018). Degree awareness (Pei et al. 2019) was proposed to mitigate the impact of degree difference in the task of knowledge graph alignment in which entities with similar degrees in knowledge graphs tend to be mapped into the same region in the embedding space. Another work (Zhang et al. 2021) investigates the problem of high variances and limited theoretical guarantees for sampling approaches in improving the scalability of GNNs. These works aim to address a different problem from ours.

References

Beutel, A.; Chen, J.; Doshi, T.; Qian, H.; Wei, L.; Wu, Y.; Heldt, L.; Zhao, Z.; Hong, L.; Chi, E. H.; et al. 2019. Fairness in recommendation ranking through pairwise comparisons. In *KDD*, 2212–2220.

Bose, A.; and Hamilton, W. 2019. Compositional fairness constraints for graph embeddings. In *ICML*, 715–724.

Burke, R.; Sonboli, N.; and Ordonez-Gauger, A. 2018. Balanced neighborhoods for multi-sided fairness in recommendation. In *FAccT*, 202–214.

Cai, H.; Zheng, V. W.; and Chang, K. C.-C. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *TKDE*, 1616–1637.

Dai, E.; and Wang, S. 2021. Say No to the Discrimination: Learning Fair Graph Neural Networks with Limited Sensitive Attribute Information. In *WSDM*, 680–688.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *KDD*, 855–864.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NeurIPS*, 1024–1034.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. *ICLR*.

Kojaku, S.; Yoon, J.; Constantino, I.; and Ahn, Y.-Y. 2021. Residual2Vec: Debiasing graph embedding with random graphs. In *NeurIPS*.

Li, P.; Wang, Y.; Zhao, H.; Hong, P.; and Liu, H. 2021. On dyadic fairness: Exploring and mitigating bias in graph connections. In *ICLR*.

Liu, Z.; Nguyen, T.-K.; and Fang, Y. 2021. Tail-GNN: Tail-Node Graph Neural Networks. In *KDD*, 1109–1119.

Ma, J.; Chang, B.; Zhang, X.; and Mei, Q. 2021. CopulaGNN: Towards Integrating Representational and Correlational Roles of Graphs in Graph Neural Networks. In *ICLR*.

Mehrotra, R.; McInerney, J.; Bouchard, H.; Lalmas, M.; and Diaz, F. 2018. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. In *CIKM*, 2243–2251.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, 3111–3119.

Pei, H.; Wei, B.; Chang, K. C.-C.; Lei, Y.; and Yang, B. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *ICLR*.

Pei, S.; Yu, L.; Hoehndorf, R.; and Zhang, X. 2019. Semi-supervised entity alignment via knowledge graph embedding with awareness of degree difference. In *WWW*, 3130–3136.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *KDD*, 701–710.

Rahman, T. A.; Surma, B.; Backes, M.; and Zhang, Y. 2019. Fairwalk: Towards Fair Graph Embedding. In *IJCAI*, 3289–3295.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *WWW*, 1067–1077.

Tang, X.; Yao, H.; Sun, Y.; Wang, Y.; Tang, J.; Aggarwal, C.; Mitra, P.; and Wang, S. 2020. Investigating and Mitigating Degree-Related Biases in Graph Convoltuional Networks. In *CIKM*, 1435–1444.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph attention networks. *ICLR*.

Wang, Y.; Zhao, Y.; Dong, Y.; Chen, H.; Li, J.; and Derr, T. 2022. Improving Fairness in Graph Neural Networks via Mitigating Sensitive Attribute Leakage. In *SIGKDD*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How powerful are graph neural networks? *ICLR*.

Zhang, Q.; Wipf, D.; Gan, Q.; and Song, L. 2021. A biased graph neural network sampler with near-optimal regret. In *NeurIPS*.