Unbiased Graph Embedding with Biased Graph Observations
Nan Wang, Lu Lin, Jundong Li, Hongning Wang

[Equal contribution between the first two authors.]

University of Virginia
Charlottesville, VA, USA

ABSTRACT
Graph embedding techniques are pivotal in real-world machine learning tasks that operate on graph-structured data, such as social recommendation and protein structure modeling. Embeddings are mostly performed on the node level for learning representations of each node. Since the formation of a graph is inevitably affected by certain sensitive node attributes, the node embeddings can inherit such sensitive information and introduce undesirable biases in downstream tasks. Most existing works impose ad-hoc constraints on the node embeddings to restrict their distributions for unbiasedness/fairness, which however compromise the utility of the resulting embeddings. In this paper, we propose a principled new way for unbiased graph embedding by learning node embeddings from an underlying bias-free graph, which is not influenced by sensitive node attributes. Motivated by this new perspective, we propose two complementary methods for uncovering such an underlying graph, with the goal of introducing minimum impact on the utility of the embeddings. Both our theoretical justification and extensive experimental comparisons against state-of-the-art solutions demonstrate the effectiveness of our proposed methods.

CCS CONCEPTS
• Computing methodologies → Machine learning; • Applied computing → Law, social and behavioral sciences.

KEYWORDS
unbiased graph embedding, sensitive attributes, bias-free graph

ACM Reference Format:
Nan Wang, Lu Lin, Jundong Li, Hongning Wang. 2022. Unbiased Graph Embedding with Biased Graph Observations. In Proceedings of the ACM Web Conference 2022 (WWW ’22), April 25–29, 2022, Virtual Event, Lyon, France. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3485447.3512189

1 INTRODUCTION
Graph embedding is an indispensable building block in modern machine learning approaches that operate on graph-structured data [12, 13, 20, 35, 41]. Graph embedding methods map each node to a low-dimensional embedding vector that reflects the nodes’ structural information from the observed connections in the given graph.

These node embeddings are then employed to solve downstream tasks, such as friend recommendation in social networks (i.e., link prediction) or user interest prediction in e-commerce platforms (i.e., node classification) [32, 44]. However, the observed node connections in a graph are inevitably affected by certain sensitive node attributes (e.g., gender, age, race, religion, etc., of users) [36], which are intended to be withheld from many high-stake real-world applications. Without proper intervention, the learned node embeddings can inherit undesired sensitive information and lead to severe bias and fairness concerns in downstream tasks [5, 37]. For example, in social network recommendation, if the users with the same gender are observed to connect more often, the learned embeddings can record such information and lead to gender bias by only recommending friends to a user with the same gender identity. Biased node embeddings, when applied in applications such as loan application [22] or criminal justice [4], may unintentionally favor or disregard one demographic group, causing unfair treatments. Besides, from the data privacy perspective, this also opens up the possibility for extraction attacks from the node embeddings [39]. These realistic and ethical concerns set a higher bar for the graph embedding methods to learn both effective and unbiased embeddings.

There is rich literature in enforcing unbiasedness/fairness in algorithmic decision making, especially in classical classification problems [8, 17, 48]. Unbiased graph embedding has just started to attract research attentions in recent years. To date, the most popular recipe for unbiased graph embedding is to add adversarial regularizations to the loss function, such that the sensitive attributes cannot be predicted from the learned embeddings [1, 5, 11, 26]. For example, making a discriminator built on the node embeddings fail to predict the sensitive attributes of the nodes. However, such a regularization is only a necessary condition for debiasing node embeddings, and it usually hurts the utility of the embeddings (a trivial satisfying solution is to randomize the embeddings). Besides these regularization-based solutions, Fairwalk [37] modifies the random walk strategy in the node2vec algorithm [13] into two levels: when choosing the next node on a path, it first randomly selects a group defined by sensitive attributes, and then randomly samples a reachable node from that group. DeBayes [6] proposes to capture the sensitive information by a prior function in Conditional Network Embedding [18], such that the learned embeddings will not carry the sensitive information. Nevertheless, both Fairwalk and DeBayes are based on specific graph embedding methods; and how to generalize them to other types of graph embedding methods such as GAT [43] or SGC [46] is not obvious.

Moving beyond the existing unbiased graph embedding paradigm, in this paper, we propose a principled new framework for the purpose with theoretical justifications. Our solution is to learn node embeddings from an underlying bias-free graph whose edges...
are generated without influence from sensitive attributes. Specifically, as suggested by Pfeiffer et al. [36], the generation of a graph can be treated as a two-phase procedure. In the first phase, the nodes are connected with each other solely based on global graph structural properties, such as degree distributions, diameter, edge connectivity, clustering coefficients and etc., resulting in an underlying structural graph, free of influences from node attributes. In the second phase, the connections are re-routed by the node attributes (including both sensitive and non-sensitive attributes). For example, in a social network, users in the same age group tend to be more connected than those in different age groups, leading to the final observed graph biased by the age attribute. Hence, our debiasing principle is to filter out the influence from sensitive attributes on the underlying structural graph to create a bias-free graph (that only has non-sensitive or no attributes) from the observed graph, and then perform embedding learning on the bias-free graph.

We propose two alternative ways to uncover the bias-free graph from the given graph for learning node embeddings. The first is a weighting-based method, which reweighs the graph reconstruction based loss function with importance sampling on each edge, such that the derived loss is as calculated on the bias-free graph, in expectation. This forms a sufficient condition for learning unbiased node embeddings: when the reconstruction loss is indeed defined on the corresponding bias-free graph, the resulting node embeddings are unbiased, since the bias-free graph is independent from the sensitive attributes. The second way is via regularization, in which we require that, with and without the sensitive attributes, the probabilities of generating an edge between two nodes from their embeddings are the same. In contrast, this forms a necessary condition: when the learning happens on the bias-free graph, the resulting embeddings should not differentiate if any sensitive attributes participated in the generation of observed graph, i.e., the predicted edge generation should be independent from the sensitive attributes. These two methods are complementary and can be combined to control the trade-off between utility and unbiasedness.

Comprehensive experiments on three datasets and several backbone graph embedding models prove the effectiveness of our proposed framework. It achieves encouraging trade-off between unbiasedness and utility of the learned embeddings. Results also suggest that the embeddings from our methods can lead to fair predictions in the downstream applications. In Section 2, we discuss the related work. We introduce the notation and preliminary knowledge on unbiased graph embedding in Section 3. We formally define the underlying bias-free graph in Section 4, and propose the unbiased graph embedding methods in Section 5. We evaluate the proposed methods in Section 6 and conclude in Section 7.

2 RELATED WORK

Graph embedding aims to map graph nodes to low-dimensional vector representations such that the original graph can be reconstructed from these node embeddings. Traditional approaches include matrix factorization and spectral clustering techniques [3, 31]. Recent years have witnessed numerous successful advances in deep neural architectures for learning node embeddings. Deepwalk [35] and node2vec [13] utilize a skip-gram [28] based objective to recover the node context in random walks on a graph. Graph Convolutional Networks (GCNs) learn a node’s embedding by aggregating the features from its neighbors supervised by node/edge labels in an end-to-end manner. These techniques are widely applied in friend or content recommendation [25, 47], protein structure prediction [16], and many more.

Recent efforts on unbiased and fair graph embedding mainly focus on pre-processing, algorithmic and post-processing steps in the learning pipeline. The pre-processing solutions modify the training data to reduce the leakage of sensitive attributes [7]. Fairwalk [37] is a typical pre-processing method which modifies the sampling process of random walk on graphs by giving each group of neighboring nodes an equal chance to be chosen. However, such pre-processing may well shift the data distribution and leads the trained model to inferior accuracy and fairness measures. The post-processing methods employ discriminators to correct the learned embeddings to satisfy specific fairness constraints [14]. However, such ad-hoc post-correction is detached from model training which can heavily degrade model’s prediction quality.

Our work falls into the category of algorithmic methods, which modify the learning objective to prevent bias from the node embeddings. The most popular algorithmic solution is adding (adversarial) regularizations as constraints to filter out sensitive information [1, 5, 10]. Compositional fairness constraints [5] are realized by a composition of discriminators for a set of sensitive attributes jointly trained with the graph embedding model. Similarly, FairGNN [10] adopts a fair discriminator but focuses on debiasing with missing sensitive attribute values. Different from regularization based methods, DeBayes [6] reformulates the maximum likelihood estimation with a biased prior which absorbs the information about sensitive attributes; but this solution is heavily coupled with the specific embedding method thus is hard to generalize. Our method differs from these previous works by learning embeddings from an underlying bias-free graph. We investigate the generation of the given graph and remove the influence from sensitive attributes in the generative process to uncover a bias-free graph for graph embedding.

Generative graph models [2, 36] focus on the statistical process of graph generation by modeling the joint distributions of edges conditioned on node attributes and graph structure. For instance, Attributed Graph Model (AGM) [36] jointly models graph structure and node attributes in a two step graph generation process. AGM first exploits a structural generative graph model to compute underlying edge probabilities based on the structural properties of a given graph. It then learns attribute correlations among edges from the observed graph and combines them with the structural edge probabilities to sample edges conditioned on attribute values. This process motivates us to uncover an underlying bias-free graph by separating out sensitive attributes and only conditioning on non-sensitive attributes for calculating edge probabilities.

3 PRELIMINARIES

In this section, we first introduce our notations and general graph embedding concepts. Since the bias/fairness issues emerge most notably in prediction tasks involving humans, such as loan application or job recommendation, we will use user-related graphs as running examples to discuss our criterion for unbiased graph embedding. But we have to emphasize that this setting is only to illustrate the
concept of unbiased graph embedding; and our proposed solution can be applied to any graph data and selected sensitive attributes to avoid biases in the learned embeddings.

3.1 Notation
Let $G = (V, E, A)$ be an undirected, attributed graph with a set of $N$ nodes $V$, a set of edges $E \subseteq V \times V$, and a set of $N$ attribute vectors $A$ (one attribute vector for each node). We use $(u, v)$ to denote an edge between node $u$ and node $v$. The number of attributes on each node is $K$, and $A = \{a_1, a_2, \ldots, a_N\}$, where $a_u$ is a $K$-dimensional attribute value vector for node $u$. We assume all attributes are categorical and $S_i$ is the set of all possible values for attribute $i$. For example, if node $u$ is a user node, and the $i$-th attribute is gender with possible values $S_i = \{\text{Female}, \text{Male}, \text{Unknown}\}$, then $a_u[i] = \text{Female}$ indicates $u$ is a female. Without loss of generality, we assume the first $m$ attributes are sensitive, and $a_u[i:m]$ and $a_u[m+1]$ stands for the $m$ sensitive attributes and the rest of the attributes that are non-sensitive, respectively.

In the problem of graph embedding learning, we aim to learn an encoder $ENC : V \rightarrow \mathbb{R}^d$ that maps each node $u$ to a $d$-dimensional embedding vector $z_u = ENC(u)$. We focus on the unsupervised embedding setting which does not require node labels and the embeddings are learned via the link prediction task. In this task, a scoring function $s_{\theta}(z_u, z_v)$ with parameters $\theta$ is defined to predict the probability of an edge $(u, v) \in E$ between node $u$ and node $v$ in the given graph. The loss for learning node embeddings and parameters of the encoder and scoring function is defined by:

$$\sum_{(u, v) \in E} L_{\text{edge}}(s_{\theta}(z_u, z_v)), \quad (1)$$

where $L_{\text{edge}}$ is a per-edge loss function on $(u, v) \in E$. Such loss functions generally aim to maximize the likelihood of observed edges in the given graph, comparing to the negative samples of node pairs where edges are not observed [13, 29].

3.2 Unbiased Graph Embedding
Given a node $u$, we consider its embedding $z_u$ as unbiased with respect to an attribute $i$ if it is independent from the attribute. Prior works evaluate such unbiasedness in the learned node embeddings by their ability to predict the values of the sensitive attributes [5, 6, 33]. For example, they first train a classifier on a subset of node embeddings using their associated sensitive attribute values as labels. If the classifier cannot correctly predict the sensitive attribute values on the rest of node embeddings, one claims that the embeddings have low bias. If the prediction performance equals to that from random node embeddings, the learned embeddings are considered bias-free. In fact, such classifiers are often used as discriminators in adversarial methods where the classifier and the embeddings are learned jointly: the embeddings are pushed in directions where the classifier has low prediction accuracy [5, 26].

There are also studies that use fairness measures such as demographic parity or equalized opportunity to define the unbiasedness of learned embeddings [6, 14]. But we need to clarify that such fairness measures can only evaluate the fairness of the final prediction results for the intended downstream tasks, but cannot assess whether the embeddings are biased by, or contain any information about, sensitive attributes. In particular, fairness in a downstream task is only a necessary condition for unbiased embedding learning, not sufficient. The logic is obvious: unbiased embeddings can lead to fair prediction results as no sensitive attribute information is involved; but obtaining fairness in one task does not suggest the embeddings themselves are unbiased, e.g., those embeddings can still lead to unfair results in other tasks or even the fair results are obtained by other means, such as post-processing of the prediction results [45]. In Section 6, we will use both the prediction accuracy on sensitive attributes and fairness measures on final tasks to evaluate the effectiveness of our unbiased graph embedding methods.

4 EFFECT OF ATTRIBUTES IN GRAPH GENERATION
In this section, we discuss the generation of an observed graph by explicitly modeling the effects of node attributes in the process. In particular, we assume that there is an underlying structural graph behind an observed graph, whose edge distribution is governed by the global graph structural properties such as degree distributions, diameter, and clustering coefficients. The attributes in $A$ will modify the structural edge distribution based on effects like homophily in social networks, where links are rewired based on the attribute similarities of the individuals [23, 27]. The modified edge distribution is then used to generate the observed graph.

Formally, let $M$ be a structural generative graph model and $\Theta_M$ be the set of parameters that describe properties of the underlying structural graph. In particular, this set of parameters $\Theta_M$ is independent from node attributes in $A$. We consider the class of models that represent the set of possible edges in the graph as binary random variables $E_{uv}, u \in V, v \in V$; i.e., the event $E_{uv} = 1$ indicates $(u, v) \in E$. The model $M$ assigns a probability to $E_{uv}$ based on $\Theta_M, P_M(E_{uv} = 1|\Theta_M)$. Therefore, the edges of an underlying structural graph $G_M$ can be considered as samples from $\text{Bernoulli}(P_M(E_{uv} = 1|\Theta_M))$. There are many such structural models $M$ such as the Chung-Lu model [9] and Krozeneck Product Graph Model [24]. Note that $M$ does not consider node attributes in the generation of the structural graph.

Now we involve the attributes in the generative process. Let $C \in \{(a_i, a_j) | i \in V, j \in V\}$ be a variable indicating the attribute value combination on nodes $u$ and $v$, given the corresponding attribute values on them and the structural parameters $\Theta_M$, is $P_C(E_{uv} = 1|C = a_{uv}, \Theta_M)$, where $a_{uv} = (a_u, a_v)$ denotes the attribute value combination on nodes $u$ and $v$. Based on Bayes’ Theorem, we have

$$P_C(E_{uv} = 1|C = a_{uv}, \Theta_M) \quad (2)$$

$$= \frac{P_C(E_{uv} = 1|\Theta_M)}{P_C(C = a_{uv}|\Theta_M)} \frac{P_D(C = a_{uv}|E_{uv} = 1, \Theta_M)}{P_D(C = a_{uv}|\Theta_M)} \quad \forall u \in V, \forall v \in V,$$

---

Footnote 1: We acknowledge that there are cases where attribute values are continuous, where discretization techniques can be applied.
Thus we can rewrite Eq (2) by

$$P_0(E_{uv} = 1 | C = \tilde{a}_{uv}, \Theta_M) = P_M(E_{uv} = 1 | \Theta_M) \tilde{R}(\tilde{a}_{uv}). \quad (4)$$

where

$$\tilde{R}(\tilde{a}_{uv}) := \frac{P_0(\tilde{C} = \tilde{a}_{uv} | E_{uv} = 1, \Theta_M)}{P_0(\tilde{C} = a_{uv} | \Theta_M)}, \forall u \in \mathcal{V}, \forall v \in \mathcal{V}. \quad (5)$$

\(\tilde{C} \in \{(\tilde{a}_u, \tilde{a}_j) | i \in \mathcal{V}, j \in \mathcal{V}\}\) is the random variable for attribute value combinations without sensitive attributes, and \(P_0\) indicates the distributions used in generating \(\tilde{G}\). We name the class of methods that learn embeddings from \(\tilde{G}\) as UGE, simply for Unbiased Graph Embedding. Next we introduce two instances of UGE. The first is UGE-W, which reweights the per-edge loss such that the total loss is from \(\tilde{G}\) in expectation. The second method is UGE-R, which adds a regularization term to shape the embeddings to satisfy the properties as those directly learned from \(\tilde{G}\).

### 5.1 Weighting-Based UGE

To compose a loss based on \(\tilde{G}\), we modify the loss function in Eq (1) by reweighing the loss term on each edge as

$$L_{UGE-W}(\tilde{G}) = \sum_{(u,v) \in \tilde{E}} L_{edge}(s(\tilde{z}_u, \tilde{z}_v)) \tilde{R}(\tilde{a}_{uv}). \quad (6)$$

The following theorem shows that, in expectation, this new loss is equivalent to the loss for learning node embeddings from \(\tilde{G}\).

**Theorem 5.1.** Given a graph \(G\), and \(R(\tilde{a}_{uv})/R(a_{uv}), \forall (u,v) \in E\), \(L_{UGE-W}(G)\) is an unbiased loss with respect to \(\tilde{G}\).

**Proof.** We take expectation over the edge observations in \(\tilde{G}\) as

$$E[L_{UGE-W}(\tilde{G})]$$

$$= E\left[ \sum_{(u,v) \in \tilde{E}} L_{edge}(s(\tilde{z}_u, \tilde{z}_v)) \frac{R(\tilde{a}_{uv})}{R(a_{uv})} \cdot E_{uv} \right]$$

$$= \sum_{u \in \mathcal{V}, v \in \mathcal{V}} E_{uv} \cdot L_{edge}(s(\tilde{z}_u, \tilde{z}_v)) \cdot \frac{R(\tilde{a}_{uv})}{R(a_{uv})} \cdot P_{\tilde{C}}(E_{uv} = 1 | C = a_{uv}, \Theta_M)$$

$$= \sum_{u \in \mathcal{V}, v \in \mathcal{V}} E_{uv} \cdot L_{edge}(s(\tilde{z}_u, \tilde{z}_v)) \cdot P_{\tilde{C}}(E_{uv} = 1 | C = \tilde{a}_{uv}, \Theta_M)$$

$$= E\left[ \sum_{(u,v) \in \tilde{E}} L_{edge}(s(\tilde{z}_u, \tilde{z}_v)) \right].$$

The step marked by * uses Eq (3) and Eq (4). \(\square\)

UGE-W is closely related to the idea of importance sampling [21], which analyzes the edge distribution of the bias-free graph \(\tilde{G}\) by observations from the given graph \(G\). The only thing needed for deploying UGE-W in existing graph embedding methods is to calculate the weights \(R(\tilde{a}_{uv})/R(a_{uv})\). To estimate \(R(\tilde{a}_{uv})\), we need the estimates of \(P_{\tilde{C}}(C = a_{uv} | E_{uv} = 1, \Theta_M)\) and \(P_{\tilde{C}}(C = \tilde{a}_{uv} | \Theta_M)\).
With maximum likelihood estimates on the observed graph, we have
\[ P_\theta(C = a_{uv} | E_{uv} = 1, \Theta_M) \approx \frac{\sum_{(i,j) \in E} \mathbb{I}[a_{ij} = a_{uv}]}{|E|}. \] (8)
\[ P_\theta(C = a_{uv} | \Theta_M) \approx \frac{\sum_{i \in V, j \in V} \mathbb{I}[a_{ij} = a_{uv}]}{N^2}. \] (9)

Similarly, we can estimate \( \tilde{R}(\tilde{a}_{uv}) \) by
\[ P_\theta(\tilde{C} = \tilde{a}_{uv} | E_{uv} = 1, \Theta_M) \approx \frac{\sum_{(i,j) \in \tilde{E}} \mathbb{I}[\tilde{a}_{ij} = \tilde{a}_{uv}]}{|\tilde{E}|}. \] (10)
\[ P_\theta(\tilde{C} = \tilde{a}_{uv} | \Theta_M) \approx \frac{\sum_{i \in \tilde{V}, j \in \tilde{V}} \mathbb{I}[\tilde{a}_{ij} = \tilde{a}_{uv}]}{N^2}. \] (11)

Note that the estimation of \( P_\theta(\tilde{C} = \tilde{a}_{uv} | E_{uv} = 1, \Theta_M) \) is based on \( \tilde{E} \), which is unfortunately from the implicit bias-free graph \( \tilde{G} \) and unobservable. But we can approximate it with \( E \) in the following way: after grouping node pairs by non-sensitive attribute value combinations \( \tilde{a}_{uv} \), the sensitive attributes only re-route the edges but do not change the number of edges in each group. Thus,
\[ P_\theta(\tilde{C} = \tilde{a}_{uv} | E_{uv} = 1, \Theta_M) \approx \frac{\sum_{(i,j) \in E} \mathbb{I}[\tilde{a}_{ij} = \tilde{a}_{uv}]}{|E|}. \] (12)
\[ = \frac{\sum_{i \in V, j \in V, a_{ij} = a_{uv}} \mathbb{I}(i,j) \in \tilde{E}}{|\tilde{E}|}. \]
\[ = \frac{\sum_{i \in V, j \in V, a_{ij} = a_{uv}} \mathbb{I}(i,j) \in E}{|E|}. \]
\[ = \frac{\sum_{i \in V, j \in V, \tilde{a}_{ij} = \tilde{a}_{uv}} \mathbb{I}(i,j) \in \tilde{E}}{|\tilde{E}|}. \]
\[ = \frac{\sum_{i \in V, j \in V, \tilde{a}_{ij} = \tilde{a}_{uv}} \mathbb{I}(i,j) \in E}{|E|}. \]

For node pairs with the same attribute value combination, Eq (8)-Eq (11) only need to be calculated once instead of for each pair. This can be done by first grouping node pairs by their attribute value combinations and then perform estimation in each group. However, when there are many attributes or attributes can take many unique values, the estimates may become inaccurate since there will be many groups and each group may only have a few nodes. In this case, we can make independence assumptions among the attributes. For example, by assuming they are independent, the estimate for a specific attribute value combination over all the \( K \) attributes becomes the product of \( K \) estimates, one for each attribute. The non-sensitive attributes can be safely removed under this assumption with \( \tilde{R}(\tilde{a}_{uv}) = 1 \), and only \( R(a_{uv}) \) needs to be estimated as \( R(a_{uv}) = \prod_{i=1}^{m} R(a_{uv}(i)) \). Since UGE-W only assigns pre-computed weights to the loss, the optimization based on it will not increase the complexity of any graph embedding method.

### 5.2 Regularization-Based UGE

We propose an alternative way for UGE which adds a regularization term to the loss function that pushes the embeddings to satisfy properties required by the bias-free graph \( \tilde{G} \). Specifically, when the node embeddings are learned from \( \tilde{G} \), their produced edge distributions should be the same with and without the sensitive attributes. To enforce this condition, we need to regularize the discrepancy between \( P_\theta(E_{uv} = 1|C = a_{uv}, \Theta_M) \) and \( P_\theta(E_{uv} = 1|\tilde{C} = \tilde{a}_{uv}, \Theta_M) \) induced from the node embeddings. We can use the scores in \( sg(z_u, z_v) \) as a proxy to represent edge probability produced by the embeddings of nodes \( u \) and \( v \), i.e., high \( sg(z_u, z_v) \) indicates high probability of an edge between \( u \) and \( v \). We can measure \( P_\theta(E_{uv} = 1|C = a_{uv}, \Theta_M) \) by aggregating node pairs with the same attribute value combination to marginalize out the effect of \( \Theta_M \) and focus on the influence from attributes as
\[ Q_{a_{uv}} = \frac{1}{N_{a_{uv}}} \sum_{i \in V, j \in V} sg(z_i, z_j). \] (13)

where we use \( Q_{a_{uv}} \) to denote the approximated measure of \( P_\theta(E_{uv} = 1|C = a_{uv}, \Theta_M) \), and \( N_{a_{uv}} \) is the number of node pairs that has the attribute value combination \( a_{uv} \). For pairs with the same attribute value combination, \( Q_{a_{uv}} \) only needs to be calculated once. Similarly, \( P_\theta(E_{uv} = 1|\tilde{C} = \tilde{a}_{uv}, \Theta_M) \) can be represented by \( Q_{\tilde{a}_{uv}} \), which can be obtained by aggregating the scores over pairs with non-sensitive attribute value combination \( \tilde{a}_{uv} \). Finally, we use \( \ell_2 \) distance between \( Q_{a_{uv}} \) and \( Q_{\tilde{a}_{uv}} \) as the regularization
\[ \mathcal{L}_{UGE-R}(\tilde{G}) = \min_{(u,v) \in E} \mathcal{L}_{edge}(sg(z_u, z_v)) + \lambda \sum_{u \in V, v \in V} \|Q_{a_{uv}} - Q_{\tilde{a}_{uv}}\|_2^2, \] (14)

where \( \ell_2 \) controls the trade-off between the per-edge losses and the regularization.

In contrast to adversarial regularizations employed in prior work \([1, 5, 11, 26]\), UGE-R takes a different perspective in regularizing the discrepancy between graphs with and without sensitive attributes induced from the embeddings. All previous regularization-based methods impose the constraint on individual edges. We should note that the regularization term is summed over all node pairs, which has a complexity of \( O(N^3) \) and can be costly to calculate. But in practice, we can add the regularization by only sampling batches of node pairs in each iteration during model update, and use \( \lambda \) to compensate the strength of the regularization.

### 5.3 Combined Method

As hinted in section 1, UGE-W is a sufficient condition for unbiased graph embedding, since it directly learns node embeddings from a bias-free graph. UGE-R is a necessary condition, as it requires the learned embeddings to satisfy the properties of a bias-free graph. We can combine them to trade-off the debiasing effect and utility,
\[ \mathcal{L}_{UGE-C}(\tilde{G}) = \min_{(u,v) \in E} \mathcal{L}_{edge}(sg(z_u, z_v)) + \lambda \sum_{u \in V, v \in V} \|Q_{a_{uv}} - Q_{\tilde{a}_{uv}}\|_2^2, \] (15)

where we use \( \mathcal{L}_{UGE-C}(\tilde{G}) \) to represent the combined method. \( \mathcal{L}_{UGE-C}(\tilde{G}) \) thus can leverage the advantages of both UGE-W and UGE-R to achieve better trade-offs between the unbiasedness and the utility of node embeddings in downstream tasks.

### 6 EXPERIMENTS

In this section, we study the empirical performance of UGE on three benchmark datasets in comparison to several baselines. In particular, we apply UGE to five popularly adopted backbone graph embedding models to show its wide applicability. To evaluate the debiasing performance, the node embeddings are firstly evaluated.
Compositional per-edge losses for these embedding models. (2) Generating random walks. We extend it to GCN, GAT and SGC by jumps to a group and then sample a node from that group for generating.

Instead of randomly jumping to a neighbor node, Fairwalk firstly modifies the pre-processing of random-walk generation by grouping nodes into groups according to their sensitive attributes. Then it samples a group and then samples a node from that group for generating random walks.

Baseline. For the Pokec-z and Pokec-n datasets, we apply GCN [19], GAT [42], SGC [46] and node2vec [13] as embedding models and apply debiasing methods on top of them. For each dataset, we construct positive examples for each node by collecting all neighboring nodes as positive examples. For each node, we use positive and negative examples for training and reserve the rest 10% for testing. For MovieLens-1M, we follow common practices and use MF as the embedding model [5, 37]. We do not evaluate Fairwalk on this dataset since there is no user-user connections and Fairwalk cannot be applied. Instead, we sample the value of each embedding dimension from [0, 1].

Table 1: Statistics of evaluation graph datasets.

| Statistics | Pokec-z | Pokec-n | MovieLens-1M |
|------------|---------|---------|--------------|
| # of nodes | 67,796  | 66,569  | 9,992        |
| # of edges | 882,765 | 729,129 | 1,000,209    |
| Density    | 0.00019 | 0.00016 | 0.01002      |

Fairness Constraints (CFC) [5] is an algorithmic method, which adds an adversarial regularizer to the loss by jointly training a composition of sensitive attribute discriminators. We apply CFC to all graph embedding models and tune the weight on the regularizer, where larger weights are expected to result in embeddings with less bias but lower utility. (3) Random embeddings are considered as a bias-free baseline. We generate random embeddings by uniformly sampling the value of each embedding dimension from [0, 1].

It is worth mentioning that a recent work DeBayes [6], which is based on the conditional network embedding (CNE) [18], includes the sensitive information in a biased prior for learning unbiased node embeddings. We did not include it since it is limited to CNE and cannot be easily generalized to other graph embedding models. Besides, we found the bias prior calculation in DeBayes does not scale to large graphs where the utility of resulting node embeddings is close to random. The original paper [6] only experimented with two small graph datasets with less than 4K nodes and 100K edges. By default, UGE follows Fairwalk to debias each of the sensitive attributes separately in experiments without dependence assumption between attributes. CFC debiases all sensitive attributes jointly as suggested in the original paper.4

6.1 Setup

- **Dataset.** We use three public user-related graph datasets, Pokec-z, Pokec-n and MovieLens-1M, where the users are associated with sensitive attributes to be debiased. The statistics of these three datasets are summarized in Table 1. Pokec is an online social network in Slovakia, which contains anonymized data of millions of users [40]. Based on the provinces where users belong to, we used two sampled datasets named as Pokec-z and Pokec-n adopted from [10], which consist of users belonging to two major regions of the corresponding provinces, respectively. In both datasets, each user has a rich set of features, such as education, working field, interest, etc.; and we include gender, region and age as (sensitive) attributes whose effect will be studied in our evaluation. MovieLens-1M is a popular movie recommendation benchmark, which contains around one million user ratings on movies [15]. In our experiment, we construct a bipartite graph which consists of user and movie nodes and rating relations as edges. The dataset includes gender, occupation and age information about users, which we treat as sensitive attributes to be debiased. We do not consider movie attributes, and thus when applying UGE, only user attributes are counted for our debiasing purpose.

- **Graph embedding models.** UGE is a general recipe for learning unbiased node embeddings, and can be applied to different graph embedding models. We evaluate its effectiveness on five representative embedding models in the supervised setting with the link prediction task. GCN [19], GAT [42], SGC [46] and node2vec [13] are deep learning models, and we use dot product between two node embeddings to predict edge probability between them and apply cross-entropy loss for training. MF [30] applies matrix factorization to the adjacency matrix. Each node is represented by an embedding vector learned with pairwise logistic loss [38].

- **Baselines.** We consider three baselines for generating unbiased node embeddings. (1) **Fairwalk** [37] is based on node2vec, which modifies the pre-processing of random-walk generation by grouping neighboring nodes with their values of the sensitive attributes. Instead of randomly jumping to a neighbor node, Fairwalk firstly jumps to a group and then sample a node from that group for generating random walks. We extend it to GCN, GAT and SGC by sampling random walks of size 1 to construct the corresponding per-edge losses for these embedding models. (2) **Compositional**

---

4UGE can debias either a single attribute or multiple attributes jointly by removing one or more attributes in the bias-free graph.
Table 2: Unbiasedness evaluated by Micro-F1 on Pokec-z and Pokec-n. Bold numbers highlight the best in each row.

| Dataset | Embedding Model | Prediction Target | No Debiasing | Fairwalk | CFC | UGE-W | UGE-R | UGE-C | Random |
|---------|-----------------|-------------------|--------------|----------|-----|-------|-------|-------|--------|
| Pokec-z | GAT             | Gender (Micro-F1)  | 0.6232       | 0.6135   | 0.5840 | 0.6150 | 0.6094 | **0.5747** | 0.4921 |
|         |                 | Region (Micro-F1)  | 0.8197       | 0.8080   | 0.7217 | 0.6784 | 0.7660 | **0.6356** | 0.4966 |
|         |                 | Age (Micro-F1)     | 0.0526       | 0.0522   | 0.0498 | 0.0431 | 0.0545 | **0.0429** | 0.0007 |
| Pokec-n | node2vec        | Gender (Micro-F1)  | 0.5241       | 0.5291   | 0.5241 | 0.5187 | **0.5095** | 0.5158 | 0.5078 |
|         |                 | Region (Micro-F1)  | 0.8690       | 0.8526   | 0.8423 | 0.8158 | 0.6975 | **0.6347** | 0.4987 |
|         |                 | Age (Micro-F1)     | 0.0626       | 0.0534   | 0.0426 | 0.0305 | 0.0294 | **0.0194** | 0.0002 |

Figure 3: Visualization of embeddings learned on Pokec-n. Node color represents the region of the nodes.

Figure 2: Trade-off between the utility (by NDCG@10) and unbiasedness (by Micro-F1) of different methods. Random embeddings give the lowest Micro-F1 (green line), and no debiasing gives the best NDCG@10 (blue line). An ideal debiasing method should locate itself at the upper left corner.

score means lower bias in the embeddings. Random embeddings are expected to have the lowest Micro-F1 and embeddings without debiasing should have the highest Micro-F1. We show the results on Pokec-z with GAT as base embedding model and Pokec-n with node2vec as the base embedding model in Table 2. From the results, we see that embeddings from UGE methods always have the least bias against all baselines with respect to all sensitive attributes and datasets. This confirms the validity of learning unbiased embeddings from a bias-free graph. Besides, by combining UGE-W and UGE-R, UGE-C usually produces the best debiasing effect, which demonstrates the complementary effect of the two methods.

Besides the unbiasedness, the learned embeddings need to be effective when applied to downstream tasks. In particular, we use NDCG@10 evaluated on the link prediction task to measure the utility of the embeddings. Specifically, for each target node, we create a candidate list of 100 nodes that includes all its observed neighbor nodes in the test set and randomly sampled negative nodes. Then NDCG@10 is evaluated on this list with predicted edge probabilities from the node embeddings. Figure 2a and 2b show the unbiasedness as well as the utility of embeddings from different methods in correspondence to the two datasets and embedding models in Table 2. Figure 2c shows the results on MovieLens-1M with MF as the embedding model.

In these plots, different embedding methods are represented by different shapes in the figures, and we use different colors to differentiate UGE-W, UGE-R and UGE-C. Random embeddings do not have any bias and provide the lowest Micro-F1 (green line), while embeddings without any debiasing gives the highest NDCG@10 (blue line). To achieve the best utility-unbiasedness trade-off, an ideal debiasing method should locate itself at the upper left corner. As shown in the figures, UGE based methods achieve the most encouraging trade-offs on these two contradicting objectives in most cases. UGE-C can usually achieve better debiasing effect, without sacrificing too much utility. UGE-W and UGE-R maintain high utility but are less effective than the combined version. CFC can achieve descent unbiasedness in embeddings, but the utility is seriously compromised (such as in Pokec-z and MovieLens-1M). Fairwalk unfortunately does not present an obvious debiasing effect.

To further visualize the debiasing effect of UGE, we use t-SNE to project the node embeddings on Pokec-n to a 2-D space in Figure 3. The left plot shows the embeddings learned via GCN without debiasing, and the right plot exhibits the debiased embeddings by applying UGE-C on GCN to debias the region attribute. Node colors represent the region value. Without debiasing, the embeddings are clearly clustered to reflect the regions of nodes. With UGE-C, embeddings from different regions are blended together, showing the effect of removing the region information from the embeddings.
We study whether the debiased embeddings can lead to fairness with node2vec as the embedding model. In each plot, x-axis is the UGE-C by tuning the weight on regularization. Although UGE-W Last but not least, we study the unbiasedness-utility trade-off in 6.4 Unbiasedness-Utility Tradeoff in UGE learned node embeddings. fairness in downstream tasks by effectively eliminating bias in the predictions. This study shows UGE’s ability to achieve competitive performance on link prediction. UGE-C can achieve fairness than the baselines on both DP and EO, while maintaining age perform similarly, UGE methods can achieve significantly better performance on each of the three sensitive attributes in Pokec-n are evaluated on each of the three sensitive attributes. It is defined by the maximum difference of TPRs between different sensitive attribute value combinations. EO measures the independence between true positive rate (TPR) of predicted edges and the predictions are independent from sensitive attributes, measured by the maximum difference of prediction rates between different sensitive attribute value combinations. For both sensitive attributes. It is defined by the maximum difference of TPRs of predicted edges and sensitive attributes. It is defined by the maximum difference of TPRs between different sensitive attribute value combinations for both DP and EO. Lower values suggest better fairness. We use the exact formulation of DP and EO in [6] and use the sigmoid function to convert the edge score for a pair of nodes to a probability.

We show the results on fairness vs. utility in Figure 4, which are evaluated on each of the three sensitive attributes in Pokec-n with node2vec as the embedding model. In each plot, x-axis is the DP or EO and y-axis is the NDCG@10 on link prediction. Similar to Figure 2, the ideal debiasing methods should locate at the upper left corner. Except for EO on the age attribute where all methods performs similarly, UGE methods can achieve significantly better fairness than the baselines on both DP and EO, while maintaining competitive performance on link prediction. UGE-C can achieve the most fair predictions. This study shows UGE’s ability to achieve fairness in downstream tasks by effectively eliminating bias in the learned node embeddings.

6.4 Unbiasedness-Utility Tradeoff in UGE Last but not least, we study the unbiasedness-utility trade-off in UGE-C by tuning the weight on regularization. Although UGE-W itself can already achieve promising debiasing effect, we expect that the added regularization from UGE-R can complement it for a better trade-off. In particular, we tune the regularization weights in both CFC and UGE-C and plot Micro-F1 (x-axis) vs. NDCG@10 (y-axis) from the resulting embeddings in Figure 5. Weight values are marked on each point and also listed in Appendix A. The results are obtained on Pokec-z with GAT as the embedding model and the two figures correspond to debiasing gender and region, respectively. With the same extent of bias measured by Micro-F1, embeddings from UGE-C have a much higher utility as indicated by the vertical grids. On the other hand, embeddings from UGE-C have much less bias when the utility is the same as CFC, as indicated by horizontal grids. This experiment proves a better trade-off achieved in UGE-C, which is consistent with our designs on UGE-W and UGE-R. UGE-W learns from a bias-free graph without any constraints, and it is sufficient to achieve unbiasedness without hurting the utility of the embeddings. UGE-R constrains the embeddings to have the properties of those learned from a bias-free graph, which is necessary for the embeddings to be unbiased.

7 CONCLUSION We propose a principled new way for learning unbiased node embeddings from graphs biased by sensitive attributes. The idea is to infer a bias-free graph where the influence from sensitive attributes is removed, and then learn the node embeddings from it. This new perspective motivates our design of UGE-W, UGE-R and their combined methods UGE-C. Extensive experiment results demonstrated strong debiasing effect from UGE as well as better unbiasedness-utility trade-offs in downstream applications.

We expect the principle of UGE can inspire better future designs for learning unbiased node embeddings from bias-free graphs. For example, instead of modeling the generation process and perform debiasing statistically, we can directly generate one or multiple bias-free graphs from the underlying generative graph model, and perform graph embedding on them. The regularization UGE-R can be refined with better moment matching mechanism than minimizing the $l_2$ distance. The weights in UGE-W can be modeled and learned for better debiasing effects. Besides, it is possible and promising to directly design unbiased GNN models that directly aggregate edges based on the inferred bias-free graph.

ACKNOWLEDGMENTS This work is supported by the National Science Foundation under grant IIS-1553568, IIS-1718216, IIS-2007492, and IIS-2006844.
set of embedding models on Pokec-z. In Appendix B.2, we show a complete comparison among our proposed instances of unbiased graph embedding UGE-W, UGE-R and UGE-C. In Appendix B.3, we investigate the influence of the regularization weight on the complete set of embedding models.

B.1 Additional Analysis on Undebiasedness
Table 3 summarizes the debiased and utility performance of the proposed method and baselines when using four graph neural networks on Pokec-z. Each line of attribute prediction result is followed by the corresponding performance on link prediction. Generally, UGE-W achieves the best link prediction performance and UGE-R has better debiasing effect. Combining UGE-W with UGE-R produces UGE-C with better trade-off.

B.2 Ablation Study
Figure 7 presents the performance of three proposed model (UGE-W, UGE-R and UGE-C) applied to four graph neural networks (GAT, SGC, GCN and node2vec). We can clearly observe that in most cases UGE-R has better debiasing effect compared with UGE-W, while UGE-W can better maintain the utility for downstream link prediction task. UGE-C as the combination of them indeed makes the best of the both designs.

B.3 Unbiasedness-Utility Tradeoff in UGE
In addition to Section 6.4 where we only showed the effect of regularization weight on Pokec-z with GAT as the embedding model,
Table 3: The prediction performance of node embeddings learned on Pokec-z using four graph neural networks as embedding models. In each row, we use bold to mark the best debiasedness on attribute prediction or utility on link prediction.

| Dataset | Embedding Model | Prediction Target | No Debiasing | Fairwalk | CFC | UGE-W | UGE-R | UGE-C | Random |
|---------|-----------------|-------------------|--------------|----------|-----|------|------|------|--------|
|        | GAT             | Gender (Micro-F1) | 0.6232       | 0.6135   | 0.5840 | 0.6150 | 0.6094 | **0.5747** | 0.4921 |
|        |                 | Link (NDCG@10)   | 0.3618       | 0.3280   | 0.2757 | **0.3554** | 0.3422 | 0.3376 | 0.0570 |
|        |                 | Region (Micro-F1)| 0.8197       | 0.8080   | 0.7217 | 0.6784 | 0.7660 | **0.6356** | 0.4966 |
|        |                 | Link (NDCG@10)   | 0.3618       | 0.3287   | 0.2757 | 0.3451 | **0.3547** | 0.3098 | 0.0570 |
|        |                 | Age (Micro-F1)   | 0.0526       | 0.0522   | 0.0498 | 0.0431 | 0.0545 | **0.0429** | 0.0007 |
|        |                 | Link (NDCG@10)   | 0.3618       | 0.3122   | 0.2757 | 0.3471 | 0.3205 | **0.3718** | 0.0570 |
|        | SGC             | Gender (Micro-F1)| 0.6766       | 0.6631   | 0.6520 | 0.6822 | 0.6531 | 0.6596 | 0.4921 |
|        |                 | Link (NDCG@10)   | 0.4975       | 0.4461   | 0.4011 | **0.4938** | 0.4850 | 0.4765 | 0.0570 |
|        |                 | Region (Micro-F1)| 0.7806       | 0.7820   | 0.7150 | 0.7402 | 0.7680 | 0.7323 | 0.4966 |
|        |                 | Link (NDCG@10)   | 0.4975       | 0.4460   | 0.4011 | **0.4832** | 0.4799 | 0.4644 | 0.0570 |
|        |                 | Age (Micro-F1)   | 0.0621       | 0.0662   | 0.0654 | 0.0606 | 0.0529 | **0.0510** | 0.0007 |
|        |                 | Link (NDCG@10)   | 0.4975       | 0.4461   | 0.4011 | **0.4889** | 0.4694 | 0.4630 | 0.0570 |
|        | GCN             | Gender (Micro-F1)| 0.5532       | 0.5589   | 0.5493 | 0.5306 | 0.5301 | **0.5162** | 0.4921 |
|        |                 | Link (NDCG@10)   | 0.3865       | 0.2807   | 0.3836 | **0.3851** | 0.3727 | 0.3488 | 0.0570 |
|        |                 | Region (Micro-F1)| 0.7445       | 0.7616   | 0.7693 | 0.5800 | 0.6105 | **0.4951** | 0.4966 |
|        |                 | Link (NDCG@10)   | 0.3865       | 0.2807   | 0.3836 | **0.3836** | 0.3801 | 0.3360 | 0.3386 | 0.0570 |
|        |                 | Age (Micro-F1)   | 0.0425       | 0.0416   | 0.0391 | 0.0439 | 0.0409 | **0.0324** | 0.0007 |
|        |                 | Link (NDCG@10)   | 0.3865       | 0.2807   | 0.3836 | **0.3987** | 0.3550 | 0.3391 | 0.0570 |
|        | node2vec        | Gender (Micro-F1)| 0.5248       | 0.5347   | 0.5137 | 0.5171 | **0.4949** | 0.4982 | 0.4921 |
|        |                 | Link (NDCG@10)   | 0.5491       | 0.5120   | 0.5496 | **0.5496** | 0.5430 | 0.5463 | 0.5206 | 0.0570 |
|        |                 | Region (Micro-F1)| 0.8423       | 0.8462   | 0.8423 | 0.8012 | 0.6490 | **0.6372** | 0.4966 |
|        |                 | Link (NDCG@10)   | 0.5491       | 0.5120   | 0.5496 | **0.5496** | 0.4816 | 0.5354 | 0.4506 | 0.0570 |
|        |                 | Age (Micro-F1)   | 0.0365       | 0.0404   | 0.0365 | 0.0200 | 0.0122 | **0.0068** | 0.0007 |
|        |                 | Link (NDCG@10)   | 0.5491       | 0.5120   | 0.5496 | **0.5496** | 0.5173 | 0.5439 | 0.5002 | 0.0570 |

we now include a complete analysis on unbiasedness and utility trade-off across embedding models in Figure 6. It clearly shows a trade-off: as the weight increases, we obtain a stronger debiasing effect with a cost of the utility on link prediction.