UNDER REVIEW

RESCIENCE C

Replication / ML Reproducibility Challenge 2022

[Re] On the reproducibility of "CrossWalk: Fairness-Enhanced Node Representation Learning"

Eric Zila[1], Jonathan Gerbscheid[1], Luc Sträter[1], and Kieron Kretschmar[1]

[1]University of Amsterdam, Amsterdam, Netherlands

Edited by
(Editor)

Reviewed by
Anonymous Reviewers

Received
–

Published
–

DOI
–

Reproducibility Summary

Scope of Reproducibility — The original authors present CrossWalk, an edge-reweighting algorithm which can be used in conjunction with random walk based node representation learning methods. We validate their claims of CrossWalk being characterized by a fairness-enhancing property, meaning it significantly reduces disparity, a measure of group fairness, and performance-conserving property, meaning it has an insignificant effect on task performance.

Methodology — To perform a robust validation of the original authors' claims, we develop an independent, highly-modular code-base with complete re-implementation of the original experiments. Our design enables its use by other researchers in the future to easily run ablation experiments with different datasets, experiments, or even algorithms that can be employed in conjunction with CrossWalk. Furthermore, we create an accessible implementation of CrossWalk itself under the MIT license.

Results — Our results provide solid evidence in favor of the performance-conserving property of CrossWalk. However, we find inconclusive evidence of the fairness-enhancing property of CrossWalk, mostly due to large variation in the reproduced disparity values. On the other hand, we find additional evidence in its favor by performing an experiment portraying the influence of the hyperparameters of CrossWalk.

What was easy — The original authors provide a code-base implementing their methodology, which greatly helped us in understanding the material. Furthermore, their methodology is very modular, meaning we could test most parts of the pipeline independently.

What was difficult — The original work contains discrepancies between specification of CrossWalk in the formulas, the pseudo-code, and the code-base. Also, we were unable to reproduce results for one of the datasets because of missing data. Finally, the original implementation is inadequately documented and its execution required numerous manual steps which were non-trivial and time consuming.

Communication with original authors — To clarify some details regarding the original implementation and its structure, we reached out to the authors when beginning to reproduce their work. The authors were quick to respond and answered all of our questions.

Copyright © 2023 E. Zila et al., released under a Creative Commons Attribution 4.0 International license.
Correspondence should be addressed to Kieron Kretschmar (kieron.kretschmar@outlook.com)
The authors have declared that no competing interests exist.
Code is available at https://github.com/jonathan-gerb/crosswalk-reproduction. – SWH swh:1:dir:0010ab17932c3abd9cb892f1b92da408df43689c.
Open peer review is available at https://openreview.net/forum?id=WnaVgRhlyT.

1 Introduction

Recently, graph neural networks (GNNs) became a pivotal topic in AI research [1]. While the domain itself develops rapidly, the notion of fairness in the context of graph learning lags behind [2]. When graph-based systems neglecting fairness are deployed in the real world, protected groups may become unjustly mistreated.
FairWalk [3] falls among the earliest attempts at mitigating such issues. In particular, it provides probabilities for which edge to select when generating random walks through the graph, an essential part of many current graph learning methods. Khajehnejad et al.[2] followed up on its ideas by designing CrossWalk, a more complex strategy which considers the context of nodes when weighting edges leading towards them.
In this paper, we verify the authors' claims by re-implementing CrossWalk and reproducing their results. We provide three main contributions:

1. **Full re-implementation.** We re-implement the full pipeline from scratch. The motivation is twofold: (1) arrival to the same conclusions using two independent implementations strengthens the claims; (2) it makes building on top of the original work more easy and viable. In support of the last point, we develop an MIT-licensed package allowing the application of CrossWalk to Deep Graph Library (DGL) graphs with a single line of code.

2. **Robustness check.** In the original paper, every experiment is run 5 times and no variation measure is provided. We statistically validate the original results by running the experiments over 50 independent runs and reporting on their variability.

3. **Theoretical extension.** While implementing the package, we found problematic edge cases that were not treated in the original CrossWalk algorithm. Therefore, we dived into the theory and came up with an extended formulation with useful probabilistic properties.

2 Scope of reproducibility

The paper *CrossWalk: Fairness-Enhanced Node Representation Learning* by Khajehnejad et al.[2] contributed to the field by proposing CrossWalk, a general graph-processing method designed to enhance group fairness of algorithms based on stochastic traversal of a graph. In their paper, the authors study the application of CrossWalk to random walk based node embedding methods, namely DeepWalk [4] and Node2Vec [5]. CrossWalk is discussed in more detail in Section 3. In their abstract, the authors summarize their main claims as follows:
"*Extensive experiments show the effectiveness of our algorithm to enhance fairness in various graph algorithms, including influence maximization, link prediction and node classification in synthetic and real networks, with only a very small decrease in performance.*" [2].

More details about these experiments can be found in Section 4. We separate these claims into two sub-claims, which we wish to verify:

- **Claim 1: Fairness-enhancing property.** The authors claim that the application of CrossWalk in all the experiments referenced above leads to enhanced fairness as measured by disparity, compared to both FairWalk and no reweighting-strategy as baselines.

- **Claim 2: Performance-conserving property.** The authors claim that the application of CrossWalk in all the experiments referenced above only leads to very small decreases in performance. We interpret this as the performance of every experiment is reduced by at most 10% when applying crosswalk, compared to both FairWalk and no reweighting-strategy as baselines.

3 CrossWalk

Original motivation — As their main contribution, Khajehnejad et al.[2] introduced CrossWalk, an edge reweighting algorithm designed to enhance group fairness in tasks based on node embeddings generated by random walks through graphs.

The CrossWalk algorithm — Random walks are part of techniques to obtain node embeddings like Node2Vec and Deepwalk. In this context, an edge weight w_{uv} corresponds to the probability of choosing v as the node following u during random walks. Given groups defined by some protected characteristic of choice, such as gender or race, the method amplifies edge weights leading towards nodes that are in different groups than the source node, and towards nodes in the proximity of different groups. Consequently, random walks traverse group boundaries more often. The author's intention of applying CrossWalk in this setting is to obtain node embeddings that are more fair when evaluated in the context of a downstream task. The effect the application of CrossWalk has on node embeddings is visualized in Figure 5 in the appendix.

Motivation for extension — When reproducing the paper, we encountered disparities between the CrossWalk reweighting formula found in [2, Equation 4], the CrossWalk pseudo-algorithm found in [2, Algorithm 1], and the CrossWalk implementation available on GitHub [6]. In particular, the pseudo-code covers an additional case that is neglected in the formula. Moreover, the public implementation addresses an additional case which is not mentioned throughout the original paper. Further, implementing CrossWalk according to the original formula and pseudocode leaves room for edge cases in which a division by zero may occur.
The main motivation for proposing adaptations to the CrossWalk algorithm lie in resolving these issues and ambiguities. We hope that this helps future researchers to decide whether CrossWalk is suited for their application.

Design goals for extension — Our proposed algorithm is designed such that a) the algorithm handles edge cases gracefully and b) yields nice probabilistic properties, while c) making only minimal changes to the authors' original approach, prioritizing their algorithm as originally implemented in [6] over their pseudocode and formulas. In practice, the two versions barely differ, as the extensions are mainly about handling of edge cases. The results of our experiments confirming the similarity are in Figure 4 in the Appendix. With the introduction our changes, we increase the number of cases in which the graph obtained from CrossWalk is guaranteed to be *probabilistic*, which makes it more feasible to be applied in more areas. This guarantee removes the necessity to normalize the weights before treating them as transition probabilities when doing e.g. random walks, enabling more efficient implementations of CrossWalk based applications.
Further, the probabilistic property of a graph may be useful in applications outside of random walks discussed here. For example, future research may investigate reweighting transition probabilities in Markov chains with the CrossWalk strategy, in cases where the nodes (states) are assigned to different groups, and an increase of transitions between states of different groups is desired.

Extension scope — In the following section, we present an extended formulation of the CrossWalk algorithm to address the issues described above. For this, we extend the equations and pseudo-code and make slight notational adjustmends. Finally, we show a valuable property of the reweighting strategy.

Extension proposal — In this section we assume weighted and directed graphs. However, the definitions and results can easily be extended to undirected graphs (by assuming

each edge to have an anti-parallel counterpart), and unweighted graphs (by assuming all initial edge weights to be equal).
Let $G = (V, E, w)$ be a weighted graph, where V is the graph's node set, E is the set of edges between the nodes, and $w_{vu} > 0$ are positive edge weights for all edges $(v, u) \in E$. We define $\mathcal{N}(v)$ as the set of neighbors of node $v \in V$ for which an edge from v to u exists. We define $c(z)$ as a function returning the *color* of node $z \in V$. In the context of fairness, color refers to the group defined by the protected characteristic that z belongs to. Let $N_v^c \subseteq \mathcal{N}(v)$ be the set of neighbors of node v of color c. Then, we can also define $R_v = \{c(u) \mid u \in \mathcal{N}(v)\} \setminus c(v)$, as the set of colors appearing in the neighborhood of node v that are different from the color of v.

Definition 1 (colorfulness)**.** We define *colorfulness* of node v, a measure of proximity of v to nodes belonging to other groups, as a function $m(v) : V \to \langle 0, 1 \rangle$. Colorfulness is estimated by taking r random walks of length d from node v using the original edge weights w. It is calculated as

$$m(v) = \frac{\sum_{j=1}^{r} \sum_{u \in \mathcal{W}_v^j} \mathbb{I}[c(v) \neq c(u)]}{rd}, \tag{1}$$

where $\mathcal{W}_v^j$ is a set of nodes in the j-th random walk from v and $\mathbb{I}$ is an indicator function.

Below are our proposed equations for computing each edge's new weight with the CrossWalk algorithm. A pseudocode-based on these equations is provided in Algorithm 1 in the Appendix. Note that our extensions to the formulas proposed originally in [2, Equation 4] are marked in pink color, and are briefly commented on below.

$$n_{vu} = \begin{cases} \dfrac{w_{vu} m(u)^p}{\sum_{z \in N_v^{c(u)}} w_{vz} m(z)^p}, & \text{if } \exists z \in N_v^{c(u)} : m(z) > 0, \quad (2a) \\ \dfrac{w_{vu}}{\sum_{z \in N_v^{c(u)}} w_{vz}} & \text{otherwise.} \quad (2b) \end{cases}$$

$$w'_{vu} = \begin{cases} (1-\alpha) n_{vu}, & \text{if } c(v) = c(u) \wedge \mathcal{N}(v) \neq N_v^{c(v)}, \quad (3a) \\ \dfrac{\alpha}{|R_v|} n_{vu}, & \text{if } c(v) \neq c(u) \wedge \mathcal{N}(v) \neq \left(\bigcup_{c \in R_v} N_v^c\right), \quad (3b) \\ n_{vu}, & \text{if } c(v) = c(u) \ \wedge \ \mathcal{N}(v) = N_v^{c(v)}, \quad (3c) \\ \dfrac{1}{|R_v|} n_{vu}, & \text{if } c(v) \neq c(u) \ \wedge \ \mathcal{N}(v) = \left(\bigcup_{c \in R_v} N_v^c\right). \quad (3d) \end{cases}$$

In the code provided with the original paper, the authors added 0.001 when estimating a node's colorfulness with Equation 1. This prevented the possibility of dividing by zero in Equation 2a, but was not mentioned in the formulas or pseudocode. Instead of adding that term, we propose imposing a condition to Equation 2a and adding Equation 2b to handle these cases.
Further, we impose additional conditions to Equations 3a and 3b (marked in green) and propose Equations 3c and 3d to handle these additional cases, respectively. Equation 3c is an entirely new proposal of ours to make Theorem 1 hold, while Equation 3d was already included in the authors' original implementation and pseudocode, but was missing in the mathematical notation.

Definition 2 (probabilistic graph)**.** A weighted graph $G = (V, E, w)$ is called a *probabilistic graph* if each node's outgoing edge weights sum up to 1, i.e. $\sum_{u \in \mathcal{N}(v)} w_{vu} = 1, \ \forall v \in V$.

Theorem 1. *Let $G = (V, E, w)$ be a weighted graph with positive edge weights $w_{vu} > 0$ for all edges $(v, u) \in E$ and at least one outgoing edge per node. Let $G' = (V, E, w')$ be the graph obtained from G by updating its edge weights using the CrossWalk algorithm specified in Algorithm 1. Then G' is a probabilistic graph.*

The proof can be found in Appendix B.

4 Methodology

Given the need to extend the theoretical basis of CrossWalk (see Section 3) and to address the shortcomings of the original implementation (see Section 6), we chose to re-implement the full pipeline consisting of (1) loading a graph, (2) reweighting its edges to promote fairness, (3) learning node embeddings, and (4) performing a downstream task. A schematic overview of the pipeline is given in Figure 1. In the following, we give a brief overview over each of the steps in the pipeline, with more detailed information being available in Appendix C.

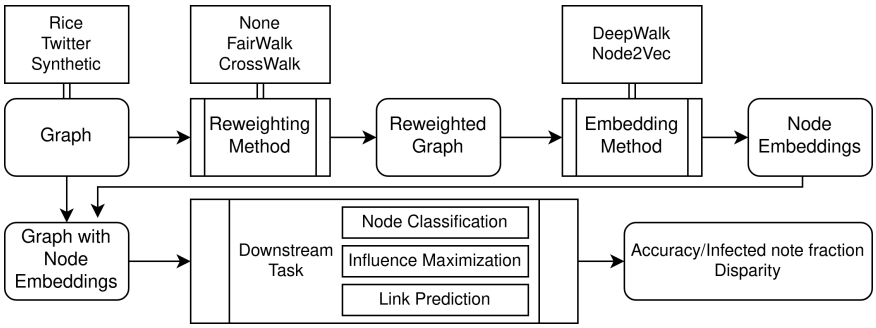


Figure 1. Schematic overview of the experimental pipeline.

4.1 Datasets

Like the original authors, we conduct experiments on two subgraphs of real social networks, and two types of synthetic graphs, which we generate anew for each experimental run to obtain more robust results. The real-world datasets are part of our publicly available codebase, as well as methods to generate synthetic ones. More information on datasets can be found in Subsection C.1.

4.2 Fairness-enhancing edge reweighting strategies

The authors propose CrossWalk as an edge reweighting strategy that can be applied to any graph with positive weights to make the embeddings generated by random walks more fair. This step of reweighting the edge weights is not strictly necessary, as uniform probabilities can also be used to generate random walks. In line with the authors, we also implement FairWalk in order to compare CrossWalk to an alternative fairness-enhancing reweighting strategy. More details about FairWalk and its differences to CrossWalk are presented in Subsection C.2.

4.3 Node embeddings

To apply machine learning algorithms on the (reweighted) graphs, one route is to learn low-dimensional node representations first and afterwards perform the downstream task on those representations. Khajehnejad et al.[2] tested CrossWalk on two random walk-based node embedding methods, namely DeepWalk and Node2Vec. These are explained in more detail in Subsection C.3

4.4 Downstream tasks and fairness evaluation

Khajehnejad et al.[2] tested CrossWalk on three common applications for learning on graphs, namely influence maximization, link prediction and node classification. These are described in more detail in Subsection C.4.

The results of each of the three tasks can be described by a global performance metric $0 \leq q \leq 1$, e.g. accuracy, and a collection of performances $Q = \{q_i\}$ evaluated on subsets of the data set defined by sensitive attributes. To obtain a notion of fairness, for each task the disparity $d = var(Q)$ is computed as the (maximum likelihood) estimate of the variance between performances on the different subsets. Finding embeddings that minimize the disparity captures the notion of fairness that these embeddings should lead to similar performances for samples of each group.

4.5 Hyperparameters

We recovered most of the original authors' hyperparameters not found in the paper by searching their codebase. Hyperparameters for the experiments on the Twitter dataset were missing (see Subsection 6.3). All of the hyperparameters we used are specified in Table 3 and Table 4, and in the configuration files in our code-base.

4.6 Experimental setup and code

The code implementing the entire pipeline as described in Section 4 is structured as Python package.[1] Each of the stages is implemeted as a submodule of the package. This simplifies the conduct of ablation studies with e.g. different datasets by future researchers. The main file of the package takes in one or more configuration files describing the experiments, calls the submodules to perform the total experiment pipeline, and writes the results to .csv files. Runs can be repeated and averaged over multiple trials. Default hyperparameters are defined in a general defaults.py file and can be overwritten by configuration files. We provide the configuration files for all experiments referenced in this paper.

4.7 Computational requirements

All experiments were ran on the CPU of a desktop with an i5-9600K @ 3.70GHz CPU and 32GB of RAM (13GB utilized). The mean runtime per experiment trial was ∼30 seconds for the Rice-Facebook and ∼13 seconds for 2- and 3-group synthetic datasets. The total runtime for reproducing all the experiments with 50 trials was ∼11 hours.

5 Results

To confirm the main claims of Khajehnejad et al.[2] presented in Section 2, we turn to the two synthetic graphs and the Rice-Facebook dataset. Overall, we find that the results reported by the original authors are statistically plausible. However, we also find that many of them are plagued by high variation of the disparity metric. In an extension of the original paper, we take notice of the importance of hyperparameter tuning when applying CrossWalk.

5.1 Results reproducing original paper

The results of the full range of experiments performed as part of the reproduction study are reported next to their counterparts from the original paper in Table 1. Additionally, we provide visualizations of our results in Figure 3 in the appendix. We successfully reproduced almost all of the experiments presented in the original paper. However, we were unable to reach comparable results for the Twitter dataset for reasons noted in Subsection 6.3. We relegate these results to Appendix F.

[1] https://github.com/jonathan-gerb/crosswalk-reproduction

Task	Dataset	Embeddings	Influence/accuracy		Disparity	
			Original	Reproduced	Original	Reproduced
IM	2-group synthetic	DW	∼12.00	10.93 (1.08)	∼48.00	44.61 (14.44)
		FW+DW	∼12.00	11.07 (0.99)	∼37.00	43.35 (14.24)
		CW+DW	∼11.00	10.27 (0.73)	∼1.00	3.04 (3.70)
	3-group synthetic	DW	∼10.00	10.56 (0.74)	∼46.00	37.59 (12.91)
		FW+DW	∼10.50	10.40 (0.77)	∼30.00	26.32 (10.20)
		CW+DW	∼10.00	10.16 (0.81)	∼16.00	9.87 (8.24)
	Rice-Facebook	DW	18.13	17.90 (2.20)	46.70	51.07 (24.64)
		FW+DW	18.20	18.47 (1.91)	20.25	25.55 (12.73)
		CW+DW	17.89	18.62 (2.22)	1.36	**22.55 (18.74)**
		N2V	18.28	18.48 (1.79)	51.88	65.54 (26.46)
		FW+N2V	18.03	18.49 (2.01)	18.29	25.93 (18.21)
		CW+N2V	18.01	17.67 (2.19)	16.87	11.33 (10.45)
NC	Rice-Facebook	DW	84.95	83.69 (3.05)	171.01	168.82 (87.15)
		FW+DW	81.94	83.12 (2.38)	71.37	72.03 (44.72)
		CW+DW	80.72	83.35 (2.72)	58.69	71.60 (54.71)
LP	Rice-Facebook	DW	76.79	**85.92 (0.53)**	38.91	**25.97 (9.83)**
		FW+DW	74.66	**84.31 (0.52)**	13.77	11.17 (3.51)
		CW+DW	73.26	**83.63 (0.48)**	3.75	**11.07 (4.06)**

Table 1. Reproduction results. Comparison of original results and their reproduced counterparts. The table contains results for influence maximization (IM), node classification (NC), and link prediction (LP). We use DeepWalk (DW) and Node2Vec (N2V) embeddings either without reweighting, with FairWalk (FW), or with CrossWalk (CW). Original results over 5 runs were gathered from the original code-base. Unavailable values (preceded by '∼') were estimated from plots in the paper. Reproduced results over 50 runs are presented with standard deviations in parentheses. For original results deviating by more than one standard deviation, the reproduced results are in **bold**.

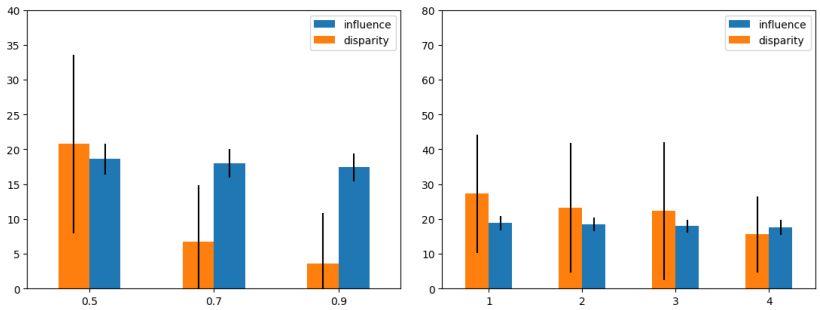
Looking at Table 1, one can see that the reproduced values generally validate those reported in the original paper. For the influence maximization and node classification tasks, all but one of the original results fall within one standard deviation of our reproduced results carried out over 50 independent runs. For the link prediction task, the differences are much starker: in terms of accuracy, we find significant improvement over the original results; in terms of disparity, we find significantly lower values when not applying any reweighting strategy and higher values when using CrossWalk.

Focusing on the performance metrics, we see no statistically significant differences between the results obtained with and without CrossWalk. This observation is in line with the one made by the original authors. **In particular, these results provide evidence in support of the performance-conserving property (Claim 2) of CrossWalk.** However, we see one exception in the case of link prediction performed on the Rice-Facebook dataset, where a significant drop in total accuracy can be observed when a fairness-enhancing reweighting strategy is applied.

Disparity is the original authors' fairness metric of choice. In general, the results presented in Table 1 show that mean disparity lowers significantly when a reweighting strategy, either FairWalk or CrossWalk, is employed. When performing influence maximization on the two synthetic datasets, we also see a significant improvement of CrossWalk over FairWalk. However, performing downstream tasks on the Rice-Facebook dataset results in statistically insignificant differences between the two, with the only large improvement margin appearing for Node2Vec embeddings. **Thus, we find inconclusive evidence of the fairness-enhancing property (Claim 1).** In particular, because of large variability of disparity results, the claim of the original authors that CrossWalk consistently provides significant improvement over FairWalk is not supported by the reproduced results.

5.2 Results beyond original paper

While we find insufficient evidence of CrossWalk outperforming FairWalk when dealing with the Rice-Facebook dataset, we argue that CrossWalk may simply require additional hyperparameter tuning. We perform a manual search for the influence maximization task on the dataset for both α and p (see Section 3). From the results in Figure 2, we observe that increasing α can greatly decrease disparity at an insignificant cost in performance. An opposite pattern emerges when we lower p from the suggested value of 4 to 1, with a lower p resulting in an increase in disparity. Clearly, it is necessary to tune both hyperparameters per experiment in order to find the desired balance between performance and disparity. Importantly, the results are in favor of the **fairness-enhancing property (Claim 1)** as with a correct choice of hyperparameters, we achieve a significant improvement over FairWalk.



(a) Hyperparameter α **(b)** Hyperparameter p

Figure 2. Impact of hyperparameters α and p (beyond the original paper). Influence and disparity for influence maximization on the Rice-Facebook dataset for different p and α values on the influence maximization task. One can observe that the trade off between accuracy and disparity can be tuned using these hyperparameters.

6 Discussion

Putting our findings presented in Section 5 into context, we believe that the main claims of the authors presented in Section 2 are supported and strengthened by our reproduction study. We find strong evidence in support of the **performance-conversion property (Claim 2)** of CrossWalk. And while we find only limited evidence in support of CrossWalk providing significant improvement in disparity over FairWalk when performing tasks on the Rice-Facebook dataset, we confirm that with additional hyperparameter tuning, the **fairness-enhancing property (Claim 1)** holds even in this case.
Moreover, we believe in our results providing strong support for CrossWalk due to our choices to (1) re-implement CrossWalk from scratch using trusted packages, such as PyTorch, DGL, and scikit-learn, thus minimizing the chances of common mistakes, (2) perform all experiments over 50 runs (rather than the original authors' 5 runs) and take into consideration the variability of the disparity measure, and (3) extend CrossWalk in order to provide better guarantees and graceful handling of more edge cases while having no big impact on the results, as can be seen in Figure 4 in the appendix.

6.1 Further work

We believe that the largest of our contributions is the highly-modular implementation. We hope that researchers will employ it for future research to easily evaluate the impact

of CrossWalk on different datasets, tasks, and fairness metrics. In the same sense, variations of CrossWalk, obtained, for example, changing the definition of colorfulness, can be implemented and extensively tested by changing just a few lines.

6.2 What was easy

The original paper is accompanied by a code-base to run all experiments. The implementation, and especially some of its unit tests, were of much help in understanding the methods in more detail. The pseudocode in the original paper was helpful, too. In contrast to FairWalk, the authors proposed CrossWalk as a reweighting method. Thereby, they decoupled it from the generation of random walks. This subtle but important change of perspective allowed for a modular implementation with higher flexibility and testability.

6.3 What was difficult

Difficulties regarding the paper — As detailed in Section 3, the CrossWalk formula, pseudocode, and actual implementation were not aligned, which made reproduction difficult. This was amplified by some notational inconsistencies. Lastly, the presentation of the results in the paper lacked exact numbers and standard deviations.

Difficulties regarding the data — The Rice-Facebook dataset in the code-base did not include all attributes needed to perform the node classification task. To perform the task, we had to track were the data came from and supply it ourselves. We could not reproduce the experiments on the Twitter dataset, because the version that was available differed from the description in the paper (see Subsection C.1 in the appendix) and because the original hyperparameters were missing. Instead, we ran the experiments on the dataset that was provided in the code-base with the hyperparameters for the Rice-Facebook dataset. The results, presented in Appendix F, did not live up to expectation. However, this may be alleviated with hyperparameter tuning.

Difficulties regarding the code — The provided code-base came without a license, hindering accessibility. Further issues include: lack of documentation, inconsistent variable naming, hard-coded hyperparameters, missing environment specifications and no seeding methods being used for reproducibility. Lastly, to re-run one experiment end-to-end with the original code-base, we had to guess the necessary yet undocumented process of identifying files containing partial results, moving them to different folders twice and copying the printed results into a notebook.

6.4 Communication with original authors

Due to the absence of an open-source license in the original code repository, we initially reached out to the original paper's authors to ensure we could use their code. Furthermore, we wished to gain insight into some of their design choices. We received a prompt response providing assurance about the code use and the needed explanations.

References

1. A. Keramatfar, M. Rafiee, and H. Amirkhani. **Graph Neural Networks: a bibliometrics overview**. 2022. DOI: 10.48550/ARXIV.2201.01188. URL: https://arxiv.org/abs/2201.01188.
2. A. Khajehnejad, M. Khajehnejad, M. Babaei, K. P. Gummadi, A. Weller, and B. Mirzasoleiman. "CrossWalk: Fairness-Enhanced Node Representation Learning." In: **Proceedings of the AAAI Conference on Artificial Intelligence** 36.11 (June 2022), pp. 11963–11970. DOI: 10.1609/aaai.v36i11.21454. URL: https://ojs.aaai.org/index.php/AAAI/article/view/21454.
3. T. Rahman, B. Surma, M. Backes, and Y. Zhang. "Fairwalk: Towards Fair Graph Embedding." In: **Proceedings of the 28th International Joint Conference on Artificial Intelligence**. IJCAI'19. Macao, China: AAAI Press, 2019, pp. 3289–3295.
4. B. Perozzi, R. Al-Rfou, and S. Skiena. "DeepWalk: Online Learning of Social Representations." In: **Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. KDD '14. New York, New York, USA: Association for Computing Machinery, 2014, pp. 701–710. DOI: 10.1145/2623330.2623732. URL: https://doi.org/10.1145/2623330.2623732.
5. A. Grover and J. Leskovec. "Node2vec: Scalable Feature Learning for Networks." In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 855–864. DOI: 10.1145/2939672.2939754. URL: https://doi.org/10.1145/2939672.2939754.
6. A. Khajehnejad, M. Khajehnejad, M. Babaei, K. P. Gummadi, A. Weller, and B. Mirzasoleiman. **CrossWalk**. 2022. URL: https://github.com/ahmadkhajehnejad/CrossWalk (visited on 01/24/2022).
7. T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space." In: **1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings**. Ed. by Y. Bengio and Y. LeCun. 2013. URL: http://arxiv.org/abs/1301.3781.
8. D. Kempe, J. Kleinberg, and E. Tardos. In: **Theory of Computing** 11.1 (2015), pp. 105–147. DOI: 10.4086/toc.2015.v011a004. URL: https://doi.org/10.4086/toc.2015.v011a004.
9. X. Zhu and G. Zoubin. **Learning from labeled and unlabeled data with label propagation**. Tech. rep. CMU-CALD-02-107. Center for Automated Learning and Discovery, School of Computer Science, Carnegie Mellon University, June 2002.

Appendix

Table of Contents

A Pseudocode for the extended CrossWalk algorithm 11

B Proof of Theorem 1 12

C Methodology - Continued 13
- C.1 Datasets - Continued . . . 13
- C.2 Fairness-enhancing edge reweighting strategies - Continued . . . 13
- C.3 Node embeddings - Continued . . . 13
- C.4 Downstream tasks and fairness evaluation - Continued . . . 14

D Hyperparameter settings 15

E Reproduction results visualized 16

F Results for Twitter dataset 17

G Other visualizations 18

A Pseudocode for the extended CrossWalk algorithm

Algorithm 1 CrossWalk algorithm

Require: Graph $G = (V, E, w)$, edge weights $w_{uv} > 0$ for all $(v, u) \in E$, parameters $\alpha \in (0, 1), p > 0$.
Ensure: Updated weights w'_{vu} for all $(v, u) \in E$.

```
for v ∈ V do                                                    ▷ Estimate colorfulness of nodes
    Run r random walks W_v^j, j ∈ {1, ..., r}, rooted at v.
    m(v) = ( Σ_{j=1}^{r} Σ_{u∈W_v^j} I[c(v) ≠ c(u)] ) / rd      ▷ [Equation 1]
end for
for v ∈ V do                                                    ▷ Iterate through all nodes
    R_v = {c(u) | u ∈ N(v)} \ c(v)
    for c ∈ {c(u) | u ∈ N(v)} do                                ▷ Iterate through all colors neighboring v
        N_v^c = {u ∈ N(v) | c(u) = c}
        for u ∈ N_v^c do                                        ▷ Iterate through v's neighbors of color c
            if ∃z ∈ N_v^c : m(z) > 0 then
                n = (w_vu m(u)^p) / ( Σ_{z∈N_v^c} w_vz m(z)^p )  ▷ [Equation 2a]
            else
                n = w_vu / Σ_{z∈N_v^c} w_vz                     ▷ [Equation 2b]
            end if
            if c = c(v) then                                    ▷ Reweight edges towards same group
                if ∃z ∈ N(v) : c(z) ≠ c(v) then
                    w'_vu = (1 − α)n                            ▷ [Equation 3a]
                else
                    w'_vu = n                                   ▷ [Equation 3c]
                end if
            else                                                ▷ Reweight edges connecting different groups
                if ∃z ∈ N(v) : c(z) = c(v) then
                    w'_vu = αn/|R_v|                            ▷ [Equation 3b]
                else
                    w'_vu = n/|R_v|                             ▷ [Equation 3d]
                end if
            end if
        end for
    end for
end for
```

B Proof of Theorem 1

Proof. For any node $v \in V$, three situations may occur. The node v may have (a) at least one same-colored and one differently-colored neighbor, (b) only neighbors of the same color, or (c) only neighbors of a different color. The possibility of nodes without outgoing edges is excluded by requirement. Further, we assume that for every neighboring group of v, there exists a neighboring node with a non-zero colorfulness, i.e. $n_{vu} = \frac{w_{vu}m(u)^p}{\sum_{z \in N_v^{c(u)}} w_{vz}m(z)^p}$ (Equation 2a). For the other cases handled by (Equation 2b), the proof follows in the exact same manner.

(a) If node v has at least one same-colored and one differently-colored neighbor, i.e. $\mathcal{N}(v) \neq N_v^{c(v)} \wedge \mathcal{N}(v) \neq \left(\bigcup_{c \in R_v} N_v^c\right)$, then $w'_{vu} = (1-\alpha)n_{vu}$ for $u \in N_v^{c(v)}$ (Equation 3a) and $w'_{vu} = \frac{\alpha}{|R_v|}n_{vu}$ for $u \in N_v^c$, $c \in R_v$ (Equation 3b)

$$\begin{aligned}
\Rightarrow \sum_{u \in \mathcal{N}(v)} w'_{vu} &= \sum_{u \in N_v^{c(v)}} (1-\alpha)\frac{w_{vu}m(u)^p}{\sum_{z \in N_v^{c(v)}} w_{vz}m(z)^p} + \sum_{c \in R_v}\sum_{u \in N_v^c} \frac{\alpha}{|R_v|}\frac{w_{vu}m(u)^p}{\sum_{z \in N_v^c} w_{vz}m(z)^p} \\
&= (1-\alpha)\frac{\sum_{u \in N_v^{c(v)}} w_{vu}m(u)^p}{\sum_{z \in N_v^{c(v)}} w_{vz}m(z)^p} + \sum_{c \in R_v} \frac{\alpha}{|R_v|}\frac{\sum_{u \in N_v^c} w_{vu}m(u)^p}{\sum_{z \in N_v^c} w_{vz}m(z)^p} \\
&= (1-\alpha) + \frac{\alpha}{|R_v|}\sum_{c \in R_v} 1 = (1-\alpha) + \frac{\alpha}{|R_v|}|R_v| = 1.
\end{aligned}$$

(b) If node v has only neighbors of the same color, i.e. $\mathcal{N}(v) = N_v^{c(v)}$, then $w'_{vu} = n_{vu}$ for $u \in N_v^{c(v)}$ (Equation 3c)

$$\Rightarrow \sum_{u \in \mathcal{N}(v)} w'_{vu} = \sum_{u \in N_v^{c(v)}} \frac{w_{vu}m(u)^p}{\sum_{z \in N_v^{c(v)}} w_{vz}m(z)^p} = \frac{\sum_{u \in N_v^{c(v)}} w_{vu}m(u)^p}{\sum_{z \in N_v^{c(v)}} w_{vz}m(z)^p} = 1.$$

(c) If node v has only neighbors different colors, i.e. $\mathcal{N}(v) = \left(\bigcup_{c \in R_v} N_v^c\right)$, then $w'_{vu} = \frac{1}{|R_v|}n_{vu}$ for $u \in N_v^c, c \in R_v$ (Equation 3d)

$$\begin{aligned}
\Rightarrow \sum_{u \in \mathcal{N}(v)} w'_{vu} &= \sum_{c \in R_v}\sum_{u \in N_v^c} \frac{1}{|R_v|}\frac{w_{vu}m(u)^p}{\sum_{z \in N_v^c} w_{vz}m(z)^p} = \sum_{c \in R_v} \frac{1}{|R_v|}\frac{\sum_{u \in N_v^c} w_{vu}m(u)^p}{\sum_{z \in N_v^c} w_{vz}m(z)^p} \\
&= \frac{1}{|R_v|}\sum_{c \in R_v} 1 = \frac{1}{|R_v|}|R_v| = 1.
\end{aligned}$$

Therefore, independent of the neighborhood of node v, it holds that $\sum_{u \in \mathcal{N}(v)} w'_{vu} = 1$ for all nodes $v \in V$, which means that G' is a probabilistic graph. □

C Methodology - Continued

C.1 Datasets - Continued

We preprocess each dataset by a) removing nodes without outgoing edges, and b) removing all edges connected to a node that is not part of the available dataset. Further, we assume all edges to be bidirectional.
A brief overview of the datasets after preprocessing can be found in Table 2. More detailed descriptions can be found in the original CrossWalk paper [2].
Note that we report roughly 5% more nodes and edges in the Twitter dataset than the original authors. We applied the original authors' preprocessing methods on the dataset we used from their code repository and have gotten the same resulting graph as in our implementation. This strengthens our belief that a) we use the same preprocessing techniques as the authors, and b) the authors used a different source dataset than the one we obtained from their code-base.

Dataset	Group 1	Group 2	Group 3	Inter-group edges
Rice-Facebook	342 (7441)	97 (513)	-	1706
Twitter†	2755 (3813)	812 (966)	186 (281)	1933
2-group synthetic	*350* (*1527*)	*150* (*279*)	-	*53*
3-group synthetic	*300* (*1121*)	*125* (*194*)	60* (*61*)	*60*

Table 2. Properties of the datasets after preprocessing. For each group the number of nodes is followed by the number of inner-group edges in parentheses. Numbers in italics are estimates, as the synthetic datasets are randomly generated.
*: This number of nodes is lower than the number reported in the original paper. The difference is due to a significant number of isolated nodes being removed in preprocessing.
†: We report roughly 5% more nodes and edges in the Twitter dataset than the original authors.

C.2 Fairness-enhancing edge reweighting strategies - Continued

FairWalk — In a similar fashion to the reproduced paper, Rahman et al.[3] proposed FairWalk to improve fairness in the context of random walk based node embeddings. When selecting a neighboring node to traverse to during random walks, FairWalk distributes equal probability of selection to all groups defined by the protected characteristic appearing in the neighborhood of the current node. Within each group, the probability is distributed equally among all neighbors, i.e., $w'_{vu} = \frac{1}{|\{c(z)|z \in \mathcal{N}(v)\}| \cdot |N_v^{c(u)}|}$.

CrossWalk vs. FairWalk — Compared to FairWalk, the reweighting strategy of CrossWalk enforces more complex rules: In addition to amplifying edges between groups (controlled by hyperparameter α), which FairWalk also achieves, CrossWalk additionally amplifies edges from nodes within a group towards the group's boundaries (controlled by hyperparameter p).

C.3 Node embeddings - Continued

DeepWalk — Perozzi, Al-Rfou, and Skiena[4] proposed a method to learn node representations, similar to how word representations are learned in Word2Vec models from natural language processing [7]. Instead of sampling sentences in which the word occurs, this method samples multiple node sequences which start from the node. These node sequences are randomly-generated walks on the graph based on the normalised edge weights and are than encoded by a number of nodes dimensional vector. This is the

input to the SkipGram model which learns a set of weights to map the random walks to a low-dimensional embedding and another set of weights to map it back to the original random walks. The output is than determined by a softmax approximating module. Lastly the difference between the input and output is used as a loss to train the weights. The learned embeddings encode how the node is connected.

Node2Vec — Grover and Leskovec[5] proposed an alteration to the DeepWalk algorithm in the way the random walks are sampled. The probability of walking to one of the neighbors is no longer only determined by the normalized edge weights, but also multiplied by a transition factor. The transition factor is $\frac{1}{p}$ when the node is the previous node in the sequence, 1 when the node is connected to the previous node in the sequence and $\frac{1}{q}$ when the node is not connected to the previous node in the sequence.

C.4 Downstream tasks and fairness evaluation - Continued

Influence maximization — The influence maximization task begins with the application of the k-medians algorithm on the node embeddings to determine k nodes which will count as initially infected in the Independent Cascades (IC) model [8] that follows. In IC, each node that was infected in the previous timestep has a chance to infect each of its neighbors that has not been infected yet. IC stops when no new nodes have been infected in a timestep. The results are q as the fraction of nodes that have been infected at the end of the experiment, and $Q = \{q_c\}$ for the fraction regarding each group c. The global performance q is the fraction of nodes that have been infected at the end of the experiment, and the grouped performances $Q = \{q_c\}$ contains those fractions regarding each group c.

Link prediction — To perform link prediction, for every node pair $(u, v) \in V \times V$ in the graph with embeddings r_u, r_v, the edge type $t((u, v)) = \{c(u), c(v)\}$ is determined as the set of their groups, and a feature vector $x((u, v)) = (r_u - r_v) \circ (r_u - r_v)$ is obtained by taking the Hadamard squared difference between their embeddings. A dataset is created by taking all node pairs that share an edge in the graph (positive samples), and randomly adding an equal number of node pairs without an edge between them (negative samples). 90% of these are selected as training data, stratified such that training and test set contain equal numbers of positive and negative samples per edge type. Finally, a logistic regression model is trained on the training set's feature vectors to predict whether the graph contains an edge between them. The global performance q is the accuracy of the model on the test set, and the grouped performances $Q = \{q_t\}$ are the accuracies for each edge type t.

Node classification — The node classification experiment is carried out on the Rice-Facebook dataset, with each node's college-id as the prediction labels. First, 50% of the nodes are randomly selected and have their labels masked. Then the Label Propagation (LP) algorithm [9] with $k = 7$ is applied on the node embeddings in an attempt to recover the masked labels. The global performance q is the accuracy with which correct labels could be recovered, and the grouped performances $Q = \{q_c\}$ contain those accuracies regarding each group c.

D Hyperparameter settings

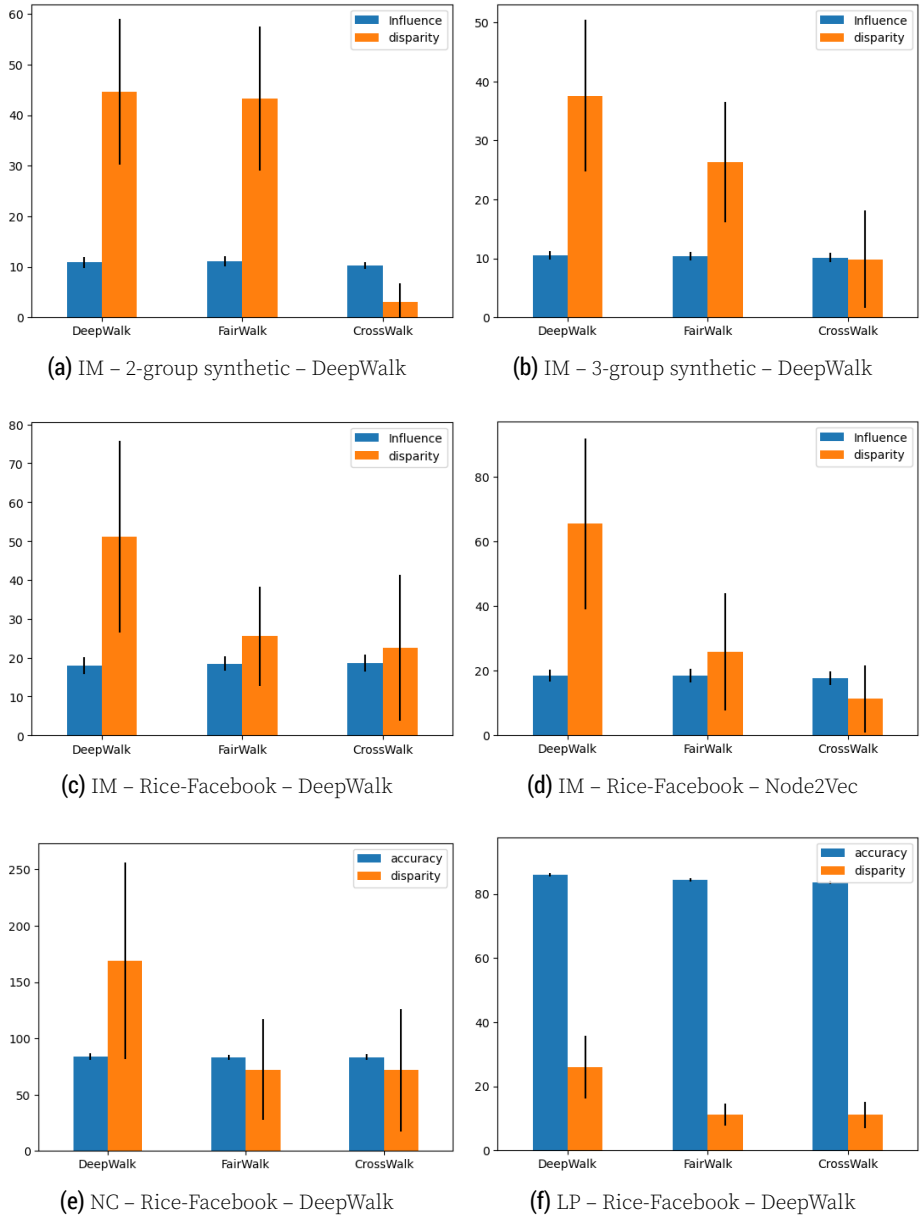
Tuning object	Parameter (symbol)	Value	Source
CrossWalk	length of random walks (d)	5	C
	number of random walks (r)	1,000	C
Node embeddings (DeepWalk + Node2Vec)	embedding space dimensionality	32	C
	length of random walks	40	C
	number of random walks per node	80	C
	context size	10	C
	negative sampling ratio	5	C
	learning rate	0.025	C
	number of epochs	5	C
Node2Vec	visited node factor denominator (p)	0.5	P
	unseen node factor denominator (q)	0.5	P
Influence maximization	number of infected nodes (k)	40	P
	infection probability		
	— synthetic graphs	0.03	P
	— real graphs	0.01	P
Node classification	test split size	0.5	P
Link prediction	test split size	0.1	P

Table 3. General hyperparameter settings. Full specification of general hyperparameter settings. These values were gathered from the original paper (P) and its public code base (C).

Downstream task	Dataset	α	p
Influence maximization	2-group synthetic	0.7	4
	3-group synthetic	0.7	4
	Rice-Facebook	0.5	4
	Twitter	0.5	2
Node classification	Rice-Facebook	0.5	2
	Twitter	0.5	2
Link prediction	Rice-Facebook	0.5	1

Table 4. CrossWalk hyperparameter settings. Full specification of CrossWalk hyperparameter settings. All of the values were gathered from the original paper.

E Reproduction results visualized



(a) IM – 2-group synthetic – DeepWalk

(b) IM – 3-group synthetic – DeepWalk

(c) IM – Rice-Facebook – DeepWalk

(d) IM – Rice-Facebook – Node2Vec

(e) NC – Rice-Facebook – DeepWalk

(f) LP – Rice-Facebook – DeepWalk

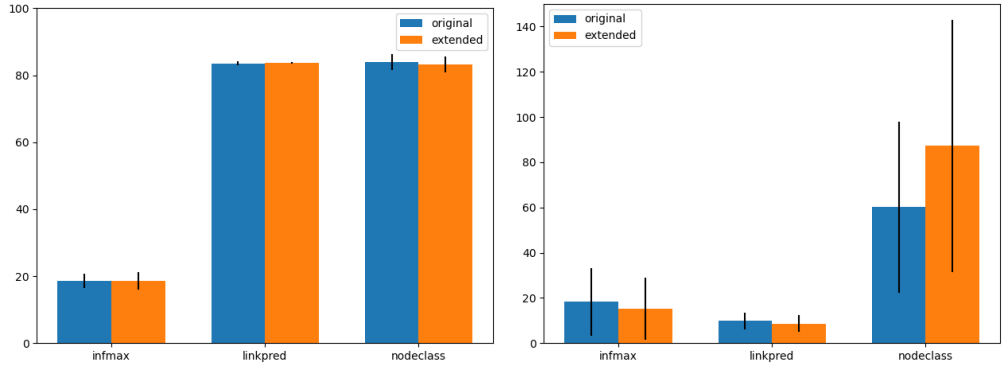
Figure 3. Visualization of reproduction results.

F Results for Twitter dataset

Task	Embeddings	Influence/accuracy		Disparity	
		Original	Reproduced	Original	Reproduced
Influence maximization	DW	∼1.37	1.30 (0.10)	∼0.09	**0.58 (0.45)**
	FW+DW	∼1.40	1.33 (0.09)	∼0.15	0.54 (0.64)
	CW+DW	∼1.43	1.33 (0.10)	∼0.08	**1.22 (0.77)**
Link prediction	DW	69.45	**92.66 (0.47)**	83.65	**3.47 (3.41)**
	FW+DW	69.17	**94.65 (0.35)**	63.82	**5.24 (5.02)**
	CW+DW	68.02	**94.66 (0.4)**	42.79	**3.93 (3.53)**

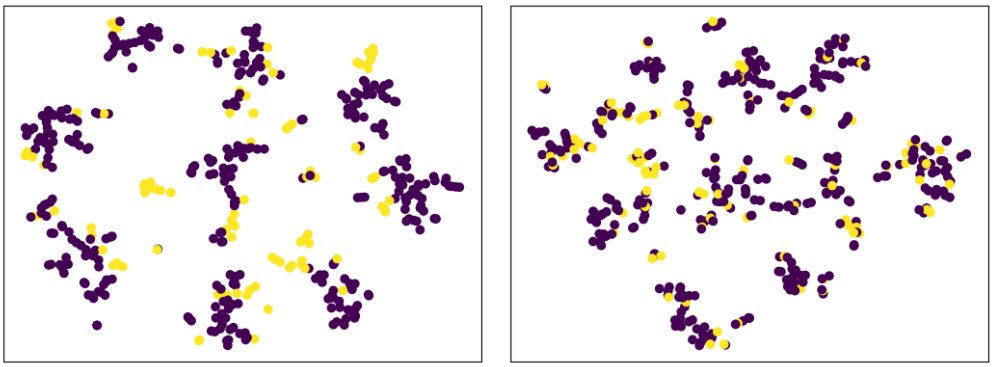
Table 5. Reproduction results for Twitter dataset. Comparison of original results and their reproduced counterparts. We use DeepWalk (DW) and Node2Vec (N2V) embeddings either without reweighting, with FairWalk (FW), or with CrossWalk (CW). Original results over 5 runs were gathered from the original code-base. Unavailable values (preceded by '∼') were estimated from plots in the paper. Reproduced results over 50 runs are presented with standard deviations in parentheses. For original results deviating by more than one standard deviation, the reproduced results are in **bold**.

G Other visualizations



(a) Performance on Rice-Facebook dataset (b) Disparity on Rice-Facebook dataset

Figure 4. Comparison of CrossWalk implementations in original formulation and proposed extended formulation (20 trials). The mean of the disparity seems higher for our extended implementation, but this very likely due to the limited amount of trials and the very large standard deviations.



(a) without reweighting (b) with CrossWalk

Figure 5. T-SNE visualization of DeepWalk embeddings of Rice-Facebook dataset without fairness-enhancing reweighting strategy and with CrossWalk. Color of the nodes represents their affiliation with groups defined by the protected characteristic. In line with the authors intentions, one can observe that embeddings of nodes from different groups appear closer together after CrossWalk has been applied.