NETEFFECT: Discovery and Exploitation of Generalized Network Effects

Meng-Chieh Lee[1(✉)], Shubhranshu Shekhar[2], Jaemin Yoo[3], and Christos Faloutsos[1]

[1] Carnegie Mellon University, Pittsburgh, USA
{mengchil,christos}@cs.cmu.edu
[2] Brandeis University, Waltham, USA
sshekhar@brandeis.edu
[3] KAIST, Seoul, South Korea
jaemin@kaist.ac.kr

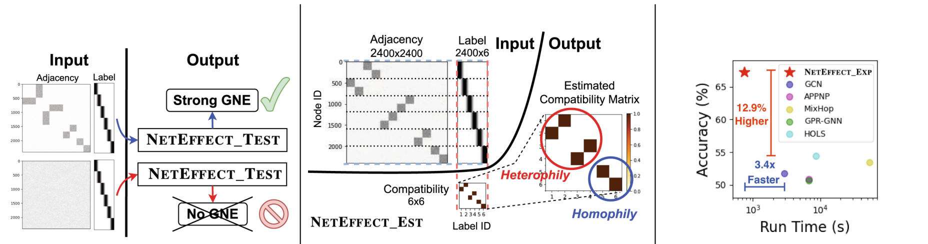
Abstract. Given a large graph with few node labels, how can we (a) identify whether there is *generalized network-effects* (GNE) or not, (b) estimate GNE to explain the interrelations among node classes, and (c) exploit GNE efficiently to improve the performance on downstream tasks? The knowledge of GNE is valuable for various tasks like node classification and targeted advertising. However, identifying GNE such as homophily, heterophily or their combination is challenging in real-world graphs due to limited availability of node labels and noisy edges. We propose NETEFFECT, a graph mining approach to address the above issues, enjoying the following properties: (i) Principled: a statistical test to determine the presence of GNE in a graph with few node labels; (ii) General and Explainable: a closed-form solution to estimate the specific type of GNE observed; and (iii) Accurate and Scalable: the integration of GNE for accurate and fast node classification. Applied on real-world graphs, NETEFFECT discovers the unexpected absence of GNE in numerous graphs, which were recognized to exhibit heterophily. Further, we show that incorporating GNE is effective on node classification. On a million-scale real-world graph, NETEFFECT achieves **over 7**$\times$ speedup (14 *minutes* vs. 2 hours) compared to most competitors.

Keywords: Network Effects · Heterophily Graphs · Node Classification

1 Introduction

Given a large graph with few node labels and no node features, how to check whether the graph structure is useful for classifying nodes or not? Node classification is often employed to infer labels on large real-world graphs. Since manual labeling is expensive and time-consuming, it is common that only few node labels are available. For example, in a million-scale social network, identifying even a fraction (say 5%) of users' groups is prohibitive, limiting the application of methods that assume many labels are given. Recently, with prevalence of graphs in industry and academia alike, there is a growing need among users to know whether these graph structures provide meaningful information for inference tasks. Therefore, before investing a huge amount of time

D.-N. Yang et al. (Eds.): PAKDD 2024, LNAI 14645, pp. 299–312, 2024.
https://doi.org/10.1007/978-981-97-2242-6_24



(a) NetEffect_Test: Principled

(b) NetEffect_Est: Explainable and General

(c) NetEffect_Exp: Accurate and Scalable

Fig. 1. **NetEffect works well**, thanks to its three novel contributions: (a) NetEffect_Test statistically the existence of GNE. (b) NetEffect_Est explains the graph with the x-ophily compatibility matrix. (c) NetEffect_Exp wins and is fast.

and resources into potentially unsuccessful experiments, a preliminary test is earnestly needed.

That is to say, we want to know whether the given graph has *generalized network-effects* (GNE) or not. A graph with GNE provides meaningful information through the structure that can be used to identify the labels of nodes. For example, "talkative person tends to make friends with talkative ones" denotes homophily, while "teenagers incline to interact with the ones that have opposite gender on social media" denotes heterophily. It is thus important to distinguish which GNE the graph has, i.e., homophily, heterophily, or both (which we call "x-ophily"), if there is any. Given c classes, an intuitive way to describe GNE is via a $c \times c$ compatibility matrix, which shows the relative influence between each class pair. It can be used to explain the graph property, as well as be exploited to better assign the labels in the graph.

However, identifying GNE is commonly neglected in literature: inference-based methods assume that the relationship is given by domain experts [4,6]; most graph neural networks (GNNs) assume homophily [9,10,24]. Although some previous works [14,16,25] use homophily statistics to analyze the given graph, our work has a very different direction because of three reasons. First, they are designed to identify the absence of homophily, and thus can not clearly distinguish GNE, which includes different non-homophily cases, i.e., heterophily, both, or no GNE. Second, to compute accurate statistics, they use all the node labels in the graph, which is impractical during node classification. Finally, their analysis rely heavily on the results of GNNs, which means in addition to the graph structure, the node features also significantly influence the conclusions of GNE. In contrast, our work aims to answer 3 research questions:

RQ1. **Hypothesis Testing**: How to identify whether the given graph has GNE or not, with only few labels?

RQ2. **Estimation**: How to estimate GNE in a principled way, and explain the graph with the estimation?

RQ3. **Exploitation**: How to efficiently exploit GNE on node classification with only few labels?

We propose NETEFFECT, with 3 contributions as the corresponding solutions:

1. **Principled: NETEFFECT_TEST** uses statistical tests to decide whether GNE exists at all. Figure 1a shows how it works, and Fig. 2 shows its discovery, where many large real-world datasets known as heterophily graphs have little GNE.
2. **General and Explainable: NETEFFECT_EST** explains whether the graph is homophily, heterophily, or X-ophily by precisely estimating the compatibility matrix with the derived closed-form formula. In Fig. 1b, it explains the interrelations of classes by the estimated compatibility matrix, which implies X-ophily.
3. **Accurate and Scalable: NETEFFECT_EXP** efficiently exploits GNE to perform better in node classification. It wins in both accuracy and time on a million-scale heterophily graph "Pokec-Gender", only requiring 14 *minutes* (Fig. 1c).

Reproducibility: The code[1] and the extended version with appendix[2] are made public.

2 Background and Related Work

2.1 Background

Notation. Let G be an undirected and unweighted graph with n nodes and m edges and an adjacency matrix $\boldsymbol{A}$. Each node i has a unique label $l(i) \in \{1, 2, \ldots, c\}$, where c is the number of classes. Let $\boldsymbol{E} \in \mathbb{R}^{n \times c}$ be the initial belief matrix with the prior information, i.e., the labeled nodes. $\boldsymbol{E}_{ik} = 1$ if $l(i) = k$, and $\boldsymbol{E}_{ik} = 0$ if $l(i) \neq k$. For the nodes without labels, their entries are set to $1/c$. $\boldsymbol{H} \in \mathbb{R}^{c \times c}$ is a row-normalized compatibility matrix, where $\boldsymbol{H}_{ku}$ is the relative influence of class l on class u. The residual of a matrix around k is $\hat{\boldsymbol{Y}} = \boldsymbol{Y} - k \times \mathbf{1}$, where $\mathbf{1}$ is matrix of ones.

Belief Propagation (BP). FABP [11] and LINBP [6] accelerate BP by approximating the final belief assignment. In particular, LINBP approximates the final belief as:

$$\hat{\boldsymbol{B}} = \hat{\boldsymbol{E}} + \boldsymbol{A}\hat{\boldsymbol{B}}\hat{\boldsymbol{H}}, \tag{1}$$

where $\hat{\boldsymbol{B}}$ is a residual final belief matrix, initialized with zeros. The compatibility matrix $\boldsymbol{H}$ and initial beliefs $\boldsymbol{E}$ are centered around $1/c$ to ensure convergence. HOLS [4] is a BP-based method, which propagates the labels by weighing with higher-order cliques.

2.2 Related Work

Table 1 presents qualitative comparison of state-of-the-art approaches against our proposed NETEFFECT. Notice that only NETEFFECT fulfills all the specs.

[1] https://github.com/mengchillee/NetEffect.
[2] https://arxiv.org/abs/2301.00270.

Table 1. **NETEFFECT matches all specs**, while baselines miss one or more. '?' and 'N/A' denote unclear and not applicable.

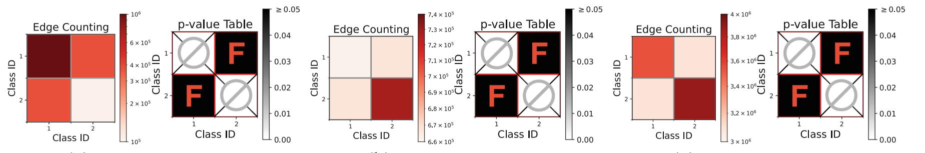
Property		BP [6,11]	HOLS [4]	General GNNs [9,10]	Het. GNNs [1,3]	NETEFFECT
1. Principled	1.1. Statistical Test	✓	✓			✓
	1.2. Convergence Guarantee	✓	✓			✓
2. Explainable	2.1 Compatibility Matrix Estimation	N/A	N/A			✓
3. General	3.1 Handle Heterophily	?	?	?	✓	✓
	3.2 Handle GNE	?	?			✓
4. Scalable	4.1. Linear Complexity	✓		✓	✓	✓
	4.2. Thrifty	✓	✓	?	?	✓

Analysis by Homophily Statistics. Many studies [14,16,25] utilize homophily ratio to measure how common the labels of the connected node pairs share the same class. Our work focuses on very different aspects, as discussed in the introduction.

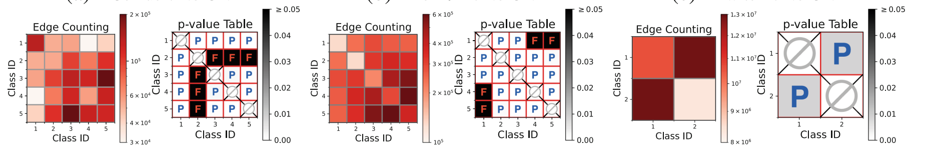
Node Classification. GCN [9] and APPNP [10] incorporate neighborhood information to do better predictions and assume homophily. MIXHOP [1], GPR-GNN [3], and H_2GCN [25] make no assumption of homophily. Nevertheless, H_2GCN requires too much memory and thus can not handle large graphs. LINKX [14] introduces multiple large heterophily datasets, but it is not applicable to graphs without node features.

3 Proposed GNE Test

Given a graph with few labels, how can we identify whether the graph has *generalized network-effects* (GNE) or not? In other words, how can we check whether the graph structure is useful for inferring node labels? We propose NETEFFECT_TEST, a statistical approach to identify the presence of GNE in a graph. Applying it to real-world graphs, we show that many popular heterophily graphs exhibit little GNE.



(a) "Genius": No GNE (b) "Penn94": No GNE (c) "Twitch": No GNE



(d) "arXiv-Year": Weak GNE (e) "Patent-Year": Weak GNE (f) "Pokec-Gender": Strong GNE

Datasets	Genius	Penn94	Twitch	Patent-Year	Pokec-Gender	arXiv-Year	Synthetic
# of Nodes / Edges / Classes	422K / 985K / 2	42KM / 1.4M / 2	168K / 6.8M / 2	1.3M / 4.3M / 5	1.6M / 22.3M / 2	169K / 1.2M / 5	1.2M / 34.0M / 6
Edge Homophily	0.618	0.470	0.545	0.132	0.425	0.222	0.314
$\hat{h}$	0.080	0.046	0.090	0.000	0.000	0.272	0.245
GNE	No GNE	No GNE	No GNE	Heterophily	Heterophily	X-ophily	X-ophily

(g) Homophily statistics of graphs and their GNE.

Fig. 2. **NETEFFECT_TEST works**: It discovers that real-world heterophily graphs do not necessarily have GNE. For each graph, we report the edge counting on the left (not available in practice), and the p-value table output from NETEFFECT_TEST on the right, where "P" denotes the presence of GNE, and "F" denotes the absence of GNE.

3.1 NETEFFECT_TEST

We first provide two main definitions regarding GNE:

Definition 1. *If the nodes with class c_i in a graph tend to connect randomly to the nodes with all classes $1, \ldots, c$ (with no specific preference), class c_i has no GNE.*

Definition 2. *If all classes in a graph have no GNE, this graph has no GNE.*

We distinguish heterophily graphs from those with no GNE by the definition. In heterophily graphs, the nodes of a specific class are likely to be connected to the nodes of other classes, such as in bipartite graphs that connect different classes of nodes. In this case, knowing the label of a node gives meaningful information about the labels of its neighbors. On the other hand, if a graph has no GNE, knowing the label of a node gives no useful information about its neighbors. In other words, the structural information of a graph is not useful to infer the unknown labels of nodes.

Next we describe how we propose to determine the existence or absence of GNE. In the inner loop, we need to decide whether class c_i (say, "talkative people"), has statistically more, or fewer edges to class c_j (say, "silent people"). We propose to use Pearson's χ^2 test for that. Specifically, given a class pair (c_i, c_j), the input to the test is a 2×2 contingency table containing the counts of edges that connect pairs of nodes whose labels are in $\{c_i, c_j\}$. The null hypothesis of the test is:

Null Hypothesis 1 *Edges are equally likely to exist between nodes of the same class and those of different classes.*

If the p-value from the test is no less than 0.05, we accept the null hypothesis, which represents that the chosen class pair (c_i, c_j) exhibits no statistically significant GNE in the graph. Then we call them *mutually indistinguishable*:

Definition 3 (Mutually indistinguishable). *Two classes c_i and c_j are* mutually indistinguishable *if we can not reject the null hypothesis above.*

Novel Implementation Details. The detailed procedure of NETEFFECT_TEST is in Appx. B.1. A practical challenge on the test is that if the numbers in the table are too large, p-value becomes very small and meaningless [15]. Uniform edge sampling can be a natural solution, but sampling for only a single round can be unstable and output very different results. To address this, we combine p-values from different random sampling by Universal Inference [23]. We firstly sample edges to add to the contingency table until the frequency is above a specified threshold, and compute the χ^2 test statistic for each class pair. Next, following Universal Inference, we repeat the procedure for random samples of edges for B rounds and average the statistics. At last, we use the average statistics to compute the p-value table $\boldsymbol{F}_{c \times c}$ of χ^2 tests. Our NETEFFECT_TEST is robust to noisy edges thanks to the sampling process, and works well given either a few or many node labels. Given a few observations, the χ^2 test works well when the frequency in the contingency table is at least 5; given many observations, our sampling trick ensures the correctness and the consistency of the computed p-value.

If a class accepts the null hypotheses with all other classes, this class has little GNE, and satisfies Def. 1. Moreover, if all classes exhibit little GNE, the whole graph satisfies Def. 2. In that case, no label propagation methods will help with node classification.

3.2 Discoveries on Real-World Graphs

We apply NetEffect_Test to 6 real-world graphs and analyze their GNE. For each dataset, we sample 5% of node labels and compute the p-value table using NetEffect_Test. This is because a) only few labels are available in most node classification tasks in practice, and thus it is reasonable to make the same assumption in the analysis, and b) NetEffect_Test can analyze GNE even from partial observations. B is set to 1000 to output stable results. Based on Def. 2 our surprising discoveries are:

Discovery 1 (No GNE) NetEffect_Test *identifies the lack of GNE in "Genius"* [13], *"Penn94"* [21], *and "Twitch"* [18]. They are widely known as heterophily graphs. In "Genius" (Fig. 2a), we see that both classes 1 and 2 tend to connect to class 1, making class 2 indistinguishable by the graph structure. NetEffect_Test thus accepts the null hypothesis, and identifies the lack of GNE. We can observe a similar phenomenon in "Penn94" (Fig. 2b). "Twitch" (Fig. 2c) used to be considered as a heterophily graph because of its weak homophily effect, but NetEffect_Test finds that each of the classes uniformly connects to both classes, and thus it has little GNE.

Discovery 2 (Heterophily and X-ophily) NetEffect_Test *identifies GNE in "Arxiv-Year", "Patent-Year", and "Pokec-Gender".* While "Patent-Year" and "Pokec-Gender" exhibit heterophily (Fig. 2e and 2f), "Arxiv-Year" exhibits X-ophily, i.e., not straight homophily or heterophily (Fig. 2d). They are thus used in our experiments.

Discovery 3 (Weak vs strong GNE) NetEffect_Test *identifies weak, and strong GNE: "Arxiv-Year" and "Patent-Year" exhibit weak GNE; and "Pokec-Gender" exhibits strong GNE.* We consider graphs to have weak GNE if there exists at least one class which is not distinguishable from some other classes. Such graphs limit the accuracy of node classification, compared with graphs with strong GNE (i.e., all classes have GNE), regardless of the specific method used for classification.

Discussion of Homophily Statistics. In Fig. 2g, we report two homophily statistics. Edge homophily [25] is the edge ratio that connect two nodes with the same class, and $\hat{h}$ [14] is an improved metric which is insensitive to the class number and size. We find even using all labels, they are not enough to capture the interrelations of all class pairs in detail, and the graphs with low homophily statistics are not guaranteed to be heterophily. They can only detect the absence of homophily, instead of distinguishing different non-homophily cases, including heterophily, X-ophily, and no GNE. In contrast, our NetEffect_Test identifies whether the graph exhibits GNE or not from only a few labels.

4 Proposed GNE Estimation

Given that a graph exhibits GNE, how can we estimate the all-pair relations between classes? A *compatibility matrix* is a natural strategy to describe the relations, which has been widely used in the literature. We propose NetEffect_Est, which turns the compatibility matrix estimation into an optimization problem based on a closed-form formula. NetEffect_Est not only overcomes the limitation of naive edge counting, but is also robust to noisy observations even with few observed labels.

4.1 Why NOT Edge Counting

The graph in Fig. 3a exhibits heterophily between class pairs $(1, 2)$ and $(3, 4)$, while it exhibits homophily in classes 5 and 6. A compatibility matrix is commonly used in existing studies, but assumed given by domain experts, instead of being estimated. A naive way to estimate it is via counting labeled edges, but it has two limitations: 1) rare labels are neglected, and 2) it is noisy or biased due to few labeled nodes. The result is even more unreliable if the given labels are imbalanced. In Fig. 3, we upsample the training labels $10\times$ for class 1 using the graph in Fig. 1b. Edge counting in Fig. 3b biases towards the upsampled class and clearly fails to estimate the correct compatibility matrix in Fig. 3a, while our proposed NETEFFECT_EST succeeds in Fig. 3c. This commonly occurs in practice, since we observe only limited labels, and becomes fatal if the observed distribution is different from the true one.

4.2 Closed-Form Formula

We begin the derivation by rewriting Eqn. 1 of BP. The main insight is reminiscent of 'leave-one-out' cross validation. That is, we find $\hat{\boldsymbol{H}}$ that would make the results of the propagation (RHS of Eqn. 2) to the actual values (LHS of Eqn. 2):

$$\underbrace{\hat{\boldsymbol{E}}}_{\text{reality}} \approx \underbrace{\boldsymbol{A}\hat{\boldsymbol{E}}\hat{\boldsymbol{H}}}_{\text{estimate}} \tag{2}$$

Formally, we want to minimize the difference between the reality and the estimate:

$$\min_{\hat{\boldsymbol{H}}} \sum_{i\in\mathcal{P}} \sum_{u=1}^{c} \|\hat{\boldsymbol{E}}_{iu} - \sum_{k=1}^{c} \sum_{j\in N(i)\cap\mathcal{P}} \hat{\boldsymbol{E}}_{jk}\hat{\boldsymbol{H}}_{ku}\|^2, \tag{3}$$

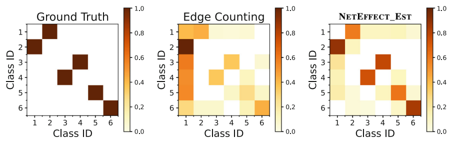
where $N(i)$ denotes the neighbors of node i. In other words, we aim to minimize the difference between initial belief $\hat{\boldsymbol{E}}$ of each node $i \in \mathcal{P}$ by the ones of its neighbors $N(i) \in \mathcal{P}$, i.e., $N(i) \cap \mathcal{P}$. To estimate the compatibility matrix $\hat{\boldsymbol{H}}$, we solve the optimization problem in Eqn. 3 with the proposed closed-form formula:

Lemma 1 (Network Effect Formula (NEF)). *Given adjacency matrix $\boldsymbol{A}$ and initial beliefs $\hat{\boldsymbol{E}}$, the closed-form solution of vectorized compatibility matrix vec$(\hat{\boldsymbol{H}})$ is:*

$$\boxed{vec(\hat{\boldsymbol{H}}) = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}} \tag{4}$$

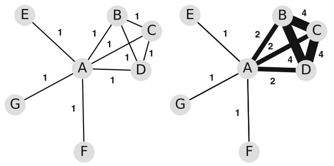
where $\boldsymbol{X} = \boldsymbol{I}_{c\times c} \otimes (\boldsymbol{A}\hat{\boldsymbol{E}})$ and $\boldsymbol{y} = vec(\hat{\boldsymbol{E}})$.

Proof. See Appx. A.1. ■



(a) Ground Truth (b) Edge Counting (c) NetEffect_Est

Fig. 3. **NetEffect_Est handles imbalanced case well**. Labels of class 1 is upsampled.



(a) Adj. Matrix (b) Emphasis Matrix

Fig. 4. **Emphasis matrix at work**: it prefers well-connected neighbors.

4.3 NetEffect_Est

The algorithm is presented in Alg. 1. In practice, we can use any form of adjacency matrix for the estimation. The proposed NEF allows us to estimate the compatibility matrix by solving this optimization problem, but there still exists a practical challenge that need to be addressed. With few labels, it is difficult to properly separate them into training and validation sets for the regression, and the estimation can easily be interfered by the noisy observations. We thus use ridge regression with leave-one-out cross-validation (RidgeCV) instead of the regular linear regression. This allows us to fully utilize the observations without having biases caused by random splits of training and validation sets. Moreover, the regularization effect of RidgeCV makes the compatibility matrix more robust to noisy observations. It is noteworthy that its computational cost is negligible.

Algorithm 1: NetEffect_Est

```
Data: Adjacency matrix $\boldsymbol{A}$, initial belief $\hat{\boldsymbol{E}}$, and priors $\mathcal{P}$
Result: Estimated compatibility matrix $\hat{\boldsymbol{H}}$
1 $\boldsymbol{X} \leftarrow \boldsymbol{I}_{c \times c} \otimes (\boldsymbol{A}\hat{\boldsymbol{E}})$;   // feature matrix
2 $\boldsymbol{y} \leftarrow \text{vec}(\hat{\boldsymbol{E}})$;   // target vector
3 Extract indices $\boldsymbol{i}$ with nodes in priors $\mathcal{P}$;
4 $\hat{\boldsymbol{H}} \leftarrow RidgeCV(\boldsymbol{X}[\boldsymbol{i}], \boldsymbol{y}[\boldsymbol{i}])$;
5 Return $\hat{\boldsymbol{H}}$;
```

5 Proposed GNE Exploitation

We propose NetEffect_Exp to exploit GNE for accurate and fast node classification with few labels. With few labels, it becomes crucial to better utilizing the graph structure. First, we address this by paying attention to influential neighbors by the proposed "emphasis" matrix; and then describe NetEffect_Exp with theoretical analysis.

5.1 "Emphasis" Matrix

Rationale and Overview. With few priors, we propose to better utilize the graph structure, by paying attention to only the most important part of it. That is to say, not all neighbors are equally influential: In Fig. 4, best practice shows that well-connected neighbors (i.e., nodes 'B', 'C', and 'D') have more influence on node 'A' than the rest.

Thus, we propose "emphasis" matrix $\boldsymbol{A}^*$, to pay attention to such neighbors. NETEFFECT_EST can also benefit from it by replacing $\boldsymbol{A}$ with $\boldsymbol{A}^*$, where we denote the improved compatibility matrix as $\hat{\boldsymbol{H}}^*$. Alg. 2 shows the details. In short, it has 3 steps:

1. **Favors influential neighbors** by quickly approximating the node-to-node proximity using (non-backtracking) random walks with restarts (lines 2-5);
2. **Touches-up** the new node-proximity by applying a series of transformations (including the best-practice element-wise logarithm) on the proximity matrix (line 6);
3. **Symmetrizes and weighs** the adjacency matrix with structural-aware embedding (lines 7-8), giving higher weights to neighbors with closer embeddings (line 9).

Algorithm 2: "Emphasis" Matrix

```
Data: Adjacency matrix $A$, number of trials $M$, number of steps $L$, and dimension $d$
Result: Emphasis matrix $A^*$
1  $W' \leftarrow O_{n\times n}$;
   /* approximate proximity matrix by random walk */
2  for node $i$ in $G$ do
3      for $m = 1, ..., M$ do
4          for $j \in \mathcal{W}_m(i, L)$ do
5              $W'_{ij} \leftarrow W'_{ij} + 1$;
   /* masking, degree normalization and logarithm */
6  $W_{n\times n} \leftarrow \log(D^{-1}(W' \odot A))$;
7  $U_{n\times d}, \Sigma_{d\times d}, V^T_{d\times n} \leftarrow \mathrm{SVD}(W, d)$;   // embedding
8  $U \leftarrow \sqrt{\Sigma}U$;   // scaling
   /* boost weights of close-embedded neighbors */
9  Weigh $A^*_{n\times n}$, where $A^*_{ij} = \mathcal{S}(U_i, U_j), \forall\{i, j | A_{ij} = 1\}$;
10 Return $A^*$;
```

Proximity Matrix Approximation. We propose to utilize random walks to approximate the proximity matrix. The approximated proximity matrix $\boldsymbol{W}'_{ij}$ records the times we visit node j if we start a random walk from node i. Only the well-connected neighbors will be visited more often. We theoretically show that it converges quickly:

Lemma 2 (Convergence of Random Walks). *With probability $1-\delta$, the error ϵ between the approximated and true distributions for a node walking to its 1-hop neighbor by random walks of length L with M trials is no greater than* $\frac{\lceil (L-1)/2 \rceil}{L}\sqrt{\frac{\log(2/\delta)}{2LM}}$.

We can make the convergence even faster by using "non-backtracking" random walks [2]. Given the start node s and walk length L, its function is defined as follows:

$$\mathcal{W}(s, L) = \left\{ (w_0 = s, ..., w_L) \begin{array}{c} w_l \in N(w_{l-1}), \forall l \in [1, L] \\ w_{l-1} \neq w_{l+1}, \forall l \in [1, L-1] \end{array} \right\}. \tag{5}$$

Thanks to it, we improve Lemma 2 to have a tighter bound of error ϵ:

Lemma 3 (Convergence of Non-backtracking Random Walks). *With the same condition as in Lemma 2, the error ϵ by non-backtracking random walks is no greater than* $\frac{\lceil (L-1)/3 \rceil}{L}\sqrt{\frac{\log(2/\delta)}{2LM}}$.

Proof. See Appx. A.2. ■

Structural-Aware Node Representation. Based on $\boldsymbol{W}$, we apply a series of transformations to generate better and unbiased representations of nodes in a fast way. An element-wise multiplication by $\boldsymbol{A}$ is done to keep the approximation of 1-hop neighbor for each node, which is sparse but supplies sufficient information. We use the inverse of the degree matrix $\boldsymbol{D}^{-1}$ to reduce the influence of nodes with large degrees. This prevents them from dominating the pairwise distance by containing more elements in their rows. The element-wise logarithm rescales the distribution in $\boldsymbol{W}$, in order to enlarge the difference between smaller structures. We use Singular Value Decomposition (SVD) for efficient rank-d decomposition of sparse $\boldsymbol{W}$, and multiply the left-singular vectors $\boldsymbol{U}$ by the squared eigenvalues $\sqrt{\boldsymbol{\Sigma}}$ to correct the scale.

"Emphasis" Matrix Construction. Directly measuring the node similarity in the graph is not trivial, or may be time consuming (e.g., by counting motifs). Therefore, we propose to compute the node similarity via the structural-aware node representations, which capture the higher-order information, and construct the "emphasis" matrix $\boldsymbol{A}^*$ by weighing $\boldsymbol{A}$ with the node similarity. The intuition is that the nodes that are closer in the embedding space are better connected with higher-order structures. The similarity function is $\mathcal{S}(\boldsymbol{U}_i, \boldsymbol{U}_j) = e^{-\mathcal{D}(\boldsymbol{U}_{ik}, \boldsymbol{U}_{jk})}$, where e is the Euler's number. It is a universal law [19], which turns the distance into similarity, and bounds it from 0 to 1. While $\mathcal{D}$ can be any distance metric, we use Euclidean as it works well empirically.

5.2 NETEFFECT_EXP

The algorithm of NETEFFECT_EXP is in Appx. B.2. NETEFFECT_EXP takes as input the "emphasis" matrix $\boldsymbol{A}^*$, the compatibility matrix $\hat{\boldsymbol{H}}^*$ estimated by $\boldsymbol{A}^*$, and the initial beliefs $\hat{\boldsymbol{E}}$. It computes the beliefs $\hat{\boldsymbol{B}}$ iteratively by aggregating the beliefs of neighbors through $\boldsymbol{A}^*$ until they converge. This reusage of $\boldsymbol{A}^*$ aims to draw attention to the neighbors that are more structurally important. By exploiting GNE with $\hat{\boldsymbol{H}}^*$, NETEFFECT propagates properly in heterophily graphs.

Convergence Guarantee. To ensure the convergence of NETEFFECT_EXP, we introduce a scaling factor f during the iterations. A smaller f leads to a faster convergence but distorts the results, thus we set f to $0.9/\rho(\boldsymbol{A}^*)$. Its exact convergence is:

Lemma 4 (Exact Convergence). *The criterion for the exact convergence of* NETEFFECT_EXP *is* $0 < f < 1/\rho(\boldsymbol{A}^*)$*, where* $\rho(\cdot)$ *denotes the spectral radius of the matrix.*

Proof. See Appx. A.3. ■

Complexity Analysis. NETEFFECT_EXP uses sparse matrix representation of graphs and scales linearly. Its complexity is:

Lemma 5. *The time complexity of* NETEFFECT_EXP *is approximately* $O(m)$ *and the space complexity is* $O(\max{(m, n \cdot L \cdot M)} + n \cdot c^2)$.

Proof. See Appx. A.4. ■

6 Experiments

In this section, we aims to answer the following questions:

Q1. **Accuracy**: How well does NETEFFECT work by estimating and exploiting GNE?
Q2. **Scalability**: How does the running time of NETEFFECT scale w.r.t. graph size?
Q3. **Explainability**: How does NETEFFECT explain the real-world graphs?

Datasets. We focus on large graphs and include 8 graphs with at least 20K nodes. For each dataset, we sample only a few node labels for training for five times and report the average. "Synthetic" is the enlarged graph in Fig. 1b, which exhibits X-ophily GNE.

Baselines. We compare NETEFFECT with five baselines and separate them into four groups: *General GNNs:* GCN [9], APPNP [10]. *Heterophily GNNs:* MIXHOP [1], GPR-GNN [3]. *BP-based methods:* HOLS [4]. *Our proposed methods:* NETEFFECT-Hom and NETEFFECT. NETEFFECT-Hom is NETEFFECT using identity matrix as compatibility matrix, which assumes homophily and does not handle GNE.

Experimental Settings. For GNNs, one-hot node degrees are used as the node features, as implemented by PyG [5]. Experiments are run on a server with 3.2GHz Intel Xeon CPU. Details of the experimental setup are in Appx. C.

Table 2. NETEFFECT wins on X-ophily and Heterophily datasets.

Dataset	Synthetic			Pokec-Gender [20]			arXiv-Year [8]			Patent-Year [12]		
# of Nodes / Edges / Classes	1.2M / 34.0M / 6			1.6M / 22.3M / 2			169K / 1.2M / 5			1.3M / 4.3M / 5		
Label Fraction	4%			0.4%			4%			4%		
GNE Strength	Strong X-ophily			Strong Heterophily			Weak X-ophily			Weak Heterophily		
Method	Accuracy	Time (s)	Rel. Time	Accuracy	Time (s)	Rel. Time	Accuracy	Time (s)	Rel. Time	Accuracy	Time (s)	Rel. Time
GCN	16.7±0.0	3456	4.1×	51.8±0.1	2906	3.4×	35.3±0.1	132	2.5×	26.0±0.0	894	2.3×
APPNP	18.6±1.1	7705	9.2×	50.9±0.3	6770	7.8×	33.5±0.2	423	8.1×	27.5±0.2	2050	5.2×
MIXHOP	16.7±0.0	58391	70.0×	53.4±1.2	53871	62.1×	39.6±0.1	2983	57.4×	26.8±0.1	18787	47.6×
GPR-GNN	18.9±1.2	7637	9.1×	50.7±0.2	6699	7.7×	30.1±1.4	400	7.7×	25.3±0.1	2034	5.1×
HOLS	46.1±0.1	1672	2.0×	54.4±0.1	8552	9.9×	34.1±0.3	566	10.9×	23.6±0.0	510	1.3×
NETEFFECT-Hom	45.6±0.1	835	1.0×	56.9±0.2	869	1.0×	37.0±0.3	52	1.0×	24.3±0.0	429	1.1×
NETEFFECT	80.4±0.0	841	1.0×	67.3±0.1	867	1.0×	38.9±0.1	52	1.0×	28.7±0.1	395	1.0×

Table 3. NETEFFECT wins on Homophily datasets.

Dataset	Facebook [17]			GitHub [17]			arXiv-Category [22]			Pokec-Locality [20]		
# of Nodes / Edges / Classes	22.5K / 171K / 4			37.7K / 289K / 2			169K / 1.2M / 40			1.6M / 22.3M / 10		
Label Fraction	4%			4%			4%			0.4%		
Method	Accuracy	Time (s)	Rel. Time	Accuracy	Time (s)	Rel. Time	Accuracy	Time (s)	Rel. Time	Accuracy	Time (s)	Rel. Time
GCN	67.0±0.8	12	2.0×	81.0±0.6	28	2.2×	25.4±0.3	216	2.3×	17.3±0.4	4002	2.9×
APPNP	50.5±2.2	46	7.7×	74.2±0.0	73	5.6×	19.4±0.6	1176	12.3×	16.8±1.7	11885	8.6×
MIXHOP	69.2±0.7	296	49.3×	77.8±1.3	526	40.5×	33.0±0.6	3203	33.4×	16.9±0.3	52139	37.9×
GPR-GNN	51.9±1.5	47	7.8×	74.1±0.1	75	5.8×	19.7±0.3	1174	12.2×	30.0±2.0	11959	8.7×
HOLS	86.0±0.4	934	155.7×	80.8±0.5	126	9.7×	61.4±0.2	627	6.5×	63.7±0.3	8139	5.9×
NETEFFECT-Hom	85.2±0.5	6	1.0×	81.3±0.5	13	1.0×	61.7±0.2	96	1.0×	66.0±0.2	1437	1.0×
NETEFFECT	85.2±0.5	6	1.0×	81.3±0.5	13	1.0×	58.8±0.6	108	1.1×	64.8±0.8	1377	1.0×

6.1 Q1 - Accuracy

In Table 2 and 3, we report the accuracy and running time. We highlight the top three from dark to light by 🟩, 🟩 and 🟩 denoting the first, second and third place. *In summary,* NETEFFECT *wins on* X*-ophily, heterophily and homophily graphs.*

X-ophily and Heterophily. In Table 2, NetEffect outperforms all the competitors significantly by more than 34.3% and 12.9% accuracy on "Synthetic" and "Pokec-Gender", respectively. These graphs exhibit strong GNE, thus NetEffect boosts the accuracy owing to precise estimations of compatibility matrix. Heterophily GNNs give results close to majority voting when the observed labels are not adequate. With homophily assumption, General GNNs and BP-based methods also not perform well. Both "arXiv-Year" and "Patent-Year" have weak GNE (Sec. 3.2). Even so, NetEffect still outperforms the competitors by estimating a reasonable compatibility matrix (Fig. 6c).

Homophily. In Table 3, NetEffect-Hom outperforms all the competitors on 3 out of 4 homophily graphs, namely "GitHub", "arXiv-Category" and "Pokec-Locality", and NetEffect performs similarly to NetEffect-Hom. In addition, NetEffect-Hom performs competitively with HOLS on "Facebook", while being 155.7× faster.

Ablation Study. *Our optimizations make a difference.* We evaluate different compatibility matrices – (i) NetEffect-EC uses edge counting on the labels of adjacent nodes in the priors, and (ii) NetEffect-A uses the adjacency matrix instead of "emphasis" matrix as the input of NetEffect_Est. To evaluate the cases when imbalanced labels are given, we upsample 5% labels to the class with the fewest labels in the datasets with weak GNE during the estimation. In Table 4, we find that NetEffect outperforms all its variants in all datasets. In the graphs with strong GNE, NetEffect shows its robustness to the structural noises and gives better results. In the imbalanced graphs, while NetEffect-EC brings its vulnerability to light, NetEffect stays with high accuracy. This study highlights the importance of a compatibility matrix estimation, as well as forming it into an optimization problem as shown in Lemma 1.

6.2 Q2 - Scalability

NetEffect *is scalable and thrifty.* We vary the edge number in "Pokec-Gender" and plot against the running time, including training and inference. In Fig. 5, NetEffect scales linearly as expected (Lemma 5). Table 5 shows the estimated AWS dollar cost in "Pokec-Gender", assuming that we use a CPU machine for NetEffect, and a GPU one for GCN. NetEffect achieves up to 45× savings. Details in Appx. C.3.

6.3 Q3 - Explainability

Figure 6 shows the compatibility matrices that NetEffect recovered. *In a nutshell, the results agree well with the intuition.* For "Synthetic", NetEffect matches the answer used for graph generation. For "Pokec-Gender" (Fig. 6a), NetEffect report heterophily, where people incline to have more opposite gender interactions [7]. For "arXiv-Year" and "Patent-Year", NetEffect find that papers and patents tend to cite the ones published in nearby years, which also agrees with intuition (Fig. 6b and 6c).

Table 4. **Ablation Study**: Estimating compatibility matrix by proposed "emphasis" matrix is essential.

Datasets	GNE Strength	NetEffect-Hom	NetEffect-EC	NetEffect-A	NetEffect
Synthetic	Strong	77.7±0.0	68.0±0.1	77.4±0.0	80.5±0.0
Pokec-Gender		56.9±0.1	64.9±0.2	64.8±0.2	67.3±0.1
arXiv-Year (imba.)	Weak	37.0±0.3	36.5±1.0	35.7±0.6	38.4±0.0
Patent-Year (imba.)		24.1±0.0	24.0±0.9	28.7±0.1	28.7±0.0

Table 5. **NetEffect is thrifty.** AWS dollar cost ($) is reported, by t3.small and p3.2xlarge.

Datasets	NetEffect	GCN
Pokec-Gender	$ 0.33 (1.0×)	$ 12.61 (45.0×)
Pokec-Locality	$ 0.53 (1.0×)	$ 13.66 (29.1×)

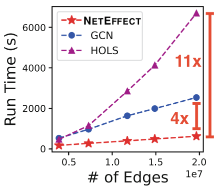


Fig. 5. **NetEffect is scalable.** It is fast and scales linearly with the edge number.

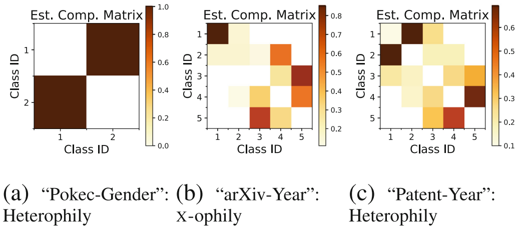


Fig. 6. **NetEffect is explainable.** Our estimated compatibility matrices are much more robust to noises compared to edge counting (in Fig. 2).

7 Conclusions

We analyze the *generalized network-effects* (GNE) in node classification in the presence of only few labels. Our proposed NetEffect has the following desirable properties:

1. *Principled*: NetEffect_Test to statistically identify the presence of GNE,
2. *General* and *Explainable*: NetEffect_Est to estimate GNE with derived closed-form solution, if there is any, and
3. *Accurate* and *Scalable*: NetEffect_Exp to efficiently exploit GNE for better performance on node classification.

Applied on a real-world graph with 22.3M edges, NetEffect only requires 14 *minutes*, and outperforms baselines on both accuracy and speed ($\geq 7\times$).

References

1. Abu-El-Haija, S., Et al. : Mixhop: higher-order graph convolutional architectures via sparsified neighborhood mixing. In: ICML, pp. 21–29 (2019)
2. Alon, N., Benjamini, I., Lubetzky, E., Sodin, S.: Non-backtracking random walks mix faster. Commun. Contemp. Math. **9**(04), 585–603 (2007)
3. Chien, E., Peng, J., Li, P., Milenkovic, O.: Adaptive universal generalized pagerank graph neural network. In: ICLR (2021)
4. Eswaran, D., Kumar, S., Faloutsos, C.: Higher-order label homogeneity and spreading in graphs. In: The Web Conference, pp. 2493–2499 (2020)

5. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)
6. Gatterbauer, W., Günnemann, S., Koutra, D., Faloutsos, C.: Linearized and single-pass belief propagation. PVLDB **8**(5), 581–592 (2015)
7. Ghosh, A., Monsivais, D., Bhattacharya, K., Dunbar, R.I., Kaski, K.: Quantifying gender preferences in human social interactions using a large cellphone dataset. EPJ Data Sci. **8**(1), 9 (2019)
8. Hu, W., et al.: Open graph benchmark: datasets for machine learning on graphs. NeurIPS **33**, 22118–22133 (2020)
9. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
10. Klicpera, J., Bojchevski, A., Günnemann, S.: Predict then propagate: Graph neural networks meet personalized pagerank. arXiv preprint arXiv:1810.05997 (2018)
11. Koutra, D., Ke, T.-Y., Kang, U., Chau, D.H., Pao, H.-K.K., Faloutsos, C.: Unifying guilt-by-association approaches: theorems and fast algorithms. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) Machine Learning and Knowledge Discovery in Databases, pp. 245–260. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23783-6_16
12. Graphs over time: densification laws, shrinking diameters and possible explanations. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp. 177–187. Association for Computing Machinery, New York (2005)
13. Lim, D., Benson, A.R.: Expertise and dynamics within crowdsourced musical knowledge curation: A case study of the genius platform. arXiv preprint arXiv:2006.08108 (2020)
14. Lim, D., et al.: Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. NeurIPS **34**, 20887–20902 (2021)
15. Lin, M., Lucas, H.C., Jr., Shmueli, G.: Research commentary-too big to fail: large samples and the p-value problem. Inf. Syst. Res. **24**(4), 906–917 (2013)
16. Ma, Y., Liu, X., Shah, N., Tang, J.: Is homophily a necessity for graph neural networks? In: ICLR (2022)
17. Rozemberczki, B., Allen, C., Sarkar, R.: Multi-scale attributed node embedding (2019)
18. Rozemberczki, B., Sarkar, R.: Twitch gamers: a dataset for evaluating proximity preserving and structural role-based node embeddings. arXiv preprint arXiv:2101.03091 (2021)
19. Shepard, R.N.: Toward a universal law of generalization for psychological science. Sci. **237**(4820), 1317–1323 (1987)
20. Takac, L., Zabovsky, M.: Data analysis in public social networks. In: International Scientific Conference and International Workshop Present Day Trends of Innovations. vol. 1 (2012)
21. Traud, A.L., Mucha, P.J., Porter, M.A.: Social structure of facebook networks. Phys. A **391**(16), 4165–4180 (2012)
22. Wang, K., Shen, Z., Huang, C., Wu, C.H., Dong, Y., Kanakia, A.: Microsoft academic graph: when experts are not enough. Quant. Sci. Stud. **1**(1), 396–413 (2020)
23. Wasserman, L., Ramdas, A., Balakrishnan, S.: Universal inference. Proc. Natl. Acad. Sci. **117**(29), 16880–16890 (2020)
24. Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., Weinberger, K.: Simplifying graph convolutional networks. In: ICML, pp. 6861–6871. PMLR (2019)
25. Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., Koutra, D.: Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. NeurIPS **34**, 7793–7804 (2020)