COST AWARE UNTARGETED POISONING ATTACK AGAINST GRAPH NEURAL NETWORKS

Yuwei Han, Yuni Lai, Yulin Zhu, Kai Zhou

Department of Computing, Hong Kong Polytechnic University, HKSAR
{22041949r, yunie.lai}@connect.polyu.hk, {yulin.zhu, kai.zhou}@polyu.edu.hk

ABSTRACT

Graph Neural Networks (GNNs) have become widely used in the field of graph mining. However, these networks are vulnerable to structural perturbations. While many research efforts have focused on analyzing vulnerability through poisoning attacks, we have identified an inefficiency in current attack losses. These losses steer the attack strategy towards modifying edges targeting misclassified nodes or resilient nodes, resulting in a waste of structural adversarial perturbation. To address this issue, we propose a novel attack loss framework called the Cost Aware Poisoning Attack (CA-attack) to improve the allocation of the attack budget by dynamically considering the classification margins of nodes. Specifically, it prioritizes nodes with smaller positive margins while postponing nodes with negative margins. Our experiments demonstrate that the proposed CA-attack significantly enhances existing attack strategies.

***Index Terms*—** Poisoning attack, graph neural networks, node classification.

1. INTRODUCTION

Graph Neural Networks (GNNs) [1–3] have emerged as an effective machine learning approach for structured data, generalizing neural networks to manage graph-structured information. They have shown potential in various applications such as social network analysis [4], anomaly detection [5–7], and natural language processing [8], excelling in tasks like node classification and link prediction. However, current research [9, 10] indicated that they are susceptible to *poisoning attacks*, where the attack easily manipulates the graph structure (i.e., adding/deleting edges) before the training of the GNN models, and leads to incorrect predictions. Such vulnerabilities not only compromise the model's reliability but also present opportunities for malicious attacks.

To analyze the vulnerabilities of GNNs, several studies have investigated potential poisoning attacks on GNNs, such as Minmax [11], Metattack [12], and certify robustness inspired attack framework [13]. These attacks leverage gradient-based techniques to approximate the complex bi-level optimization problem associated with poisoning attacks. Nonetheless, there is still potential for enhancing the current methodologies regarding the allocation of the limited perturbation budget.

We have noted Metattack with current loss functions such as negative log likelihood loss and CW loss [14] has budget waste problem that some budgets do not contribute to decrease the classification accuracy. For example, when employing the commonly used negative log likelihood loss as the attack objective, a node that has already been successfully misclassified tends to attract higher meta-gradients, resulting in a further allocation of the attack budget to that specific node. Wang et al. [13] have proposed a novel attack loss framework called the Certify Robustness Inspired Framework (CR-framework) to enhance the attack performance. This method assigns larger meta-gradients to nodes with smaller certify robustness sizes [13], as these nodes are more vulnerable to attacks. It accomplishes this by reweighting the loss of each node according to its certify robustness size. While this approach optimizes the budget allocation for existing attack losses and further decreases the accuracy of victim models, it grapples with computational inefficiencies during the assessment of certified robustness sizes.

To address the limitations above, we introduce a **Cost Aware Poisoning Attack Loss Framework (CA-attack)** to *improve the allocation of attack budget* to maximize the impact of the attack. Specifically, we dynamically reweight the nodes according to their classification margins in the attack loss. This means that the weights of the nodes are adjusted during the optimization process of the attack objective. Through extensive testing on benchmark datasets, the CA-attack consistently surpasses previous attack methods in terms of effectiveness. Our research makes the following contributions:

- To address the inefficiencies of budget allocation, we propose the CA-attack, a budget-efficient attack loss framework.
- Through rigorous empirical assessments on three datasets, we demonstrate that CA-attack improves existing methods, highlighting its potential as a plug-and-play solution for various graph poisoning attacks.

2. PRELIMINARIES

GNNs We define an undirected graph as $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_N\}$ represents the node set, and $e_{ij} \in E$ is edge connecting the nodes v_i and v_j. Node attributes are represented by $X \in \mathbb{R}^{N \times d}$, where d is the dimension of the attribute. Additionally, the graph structure can be represented by an adjacency matrix $A \in \{0, 1\}^{N \times N}$, with $A_{ij} = 1$ if an edge exists between two nodes, otherwise 0. Given a subset of labeled nodes $V_L \subset V$ where each node $v \in V_L$ has a label $y_v \in \mathcal{C} = \{c_1, c_2, \cdots, c_k\}$, GNNs aim to learn a function f_θ to predict the remaining unlabeled nodes $V_U = V \setminus V_L$ into classes of $\mathcal{C}$. We use $f_\theta(G)_v$ to denote the prediction of the model f_θ for node v. Its parameters θ are optimized by minimizing a loss $\mathcal{L}_{train}$ over the labeled nodes, typically using losses like negative log-likelihood loss or CW loss.

Attacks against GNNs Based on the classification task, the general form of node-level graph adversarial attacks can be defined as a bi-level optimization process where the attacker optimizes the allocation of the attack budget to maliciously modify the graph, while the model is optimized using this poisoned graph:

$$\begin{aligned} \min \mathcal{L}_{atk}\left(f_{\theta^*}\left(\widehat{G}\right)\right) &= \sum_{v \in V} \ell_{atk}\left(f_{\theta^*}\left(\widehat{G}\right)_v, y_v\right) \\ s.t., \theta^* &= arg\min_{\theta} \mathcal{L}_{train}\left(f_\theta\left(\widehat{G}\right)\right), \end{aligned} \tag{1}$$

where ℓ_{atk} is the attack loss and usually chosen as $\ell_{atk} = -\mathcal{L}_{train}$. $\widehat{G}$ is the graph modified from original G by the adversarial attack. y_v denotes the labels of node v.

Given a budget Δ, the attacker aims to ensure that perturbations are unnoticeable by maintaining the l_0 norm difference between the original and perturbed graph: $\|A - A'\|_0 \leq \Delta$. To avoid detection, the attacker also refrains from making significant changes to the graph's degree distribution or introducing isolated nodes, as suggested in [15].

A conventional poisoning attack process can be divided into three steps. The first step is to retrain a surrogate model $f_{\theta^*}(G)$ using a linearized graph convolutional network (GCN) which is expressed as:

$$f_\theta(G) = soft\max\left(\hat{A}^2 X W\right), \tag{2}$$

where $\hat{A} = D^{-1/2}(A + I)D^{-1/2}$ is the normalized adjacent matrix, X are the node features, D is the diagonal matrix of the node degrees, and $\theta = \{W\}$ are the set of learnable parameters. In the second step, the attacker uses pseudo-labels y_v generated by the surrogate model $f_{\theta^*}(G)$ to construct the attack loss of the node v as $\ell\left(f_{\theta^*}\left(\widehat{G}\right)_v, y_v\right)$ under the gray-box attack scenario. In the third step, the attack loss of each node is backpropagated to produce a partial gradient matrix:

$$g_{v_i} = \nabla_A \ell\left(f_{\theta^*}\left(\widehat{G}\right)_v, y_v\right), \tag{3}$$

The overall gradient information passed to the attack strategy is the average of all the partial gradient matrices. Thereafter, the attacker selects the edges to be perturbed (adding/deleting) based on the saliency of their gradients.

3. COST-AWARE ATTACK

In this section, we demonstrate our proposed CA-attack attack loss framework that aims to improve the approximation of this challenging bi-level optimization problem (1) by redesigning the $\mathcal{L}_{atk}$. Specifically, we incorporate the classification margin of nodes as dynamic weights for the attack loss.

3.1. Limitations of Previous Attacks

Since poisoning attack aims to minimize the attack loss (i.e., maximize training loss) by modifying an edge with the largest gradient at each iteration, the node connected to this edge will be attacked with higher priority according to Equation (3). Figure 1 illustrates the priority of a node in a clean graph to be attacked based on the l_2 norm of partial gradient metric for each node in Metattack using different attack loss functions. We can observe that the negative log-likelihood loss and the CR framework result in significant gradients for nodes with negative margins. Additionally, the negative log-likelihood loss produces significant gradients for nodes with large margins. In contrast, our CA framework generates significant gradients on nodes with small positive margins. We can also observe that the CW loss imparts significant gradients to nodes possessing negative margins and the CR framework exhibits difficulty concentrating on nodes with minute margins when subjected to CW loss. When applying the CA framework to the CW loss, we only assign weights to nodes with positive margins, while nodes with negative margins are given a weight of zero. In Section 4, we will demonstrate that the CA attack loss outperforms previous attack losses.

3.2. Cost Aware Loss

The classification margin of a node v is commonly defined as:

$$\varphi(v) = z_{c^*} - \max_{c \neq c^*} z_c, \tag{4}$$

where z represents the vector of logits produced by the model towards node v, and c^* refers to the true label of the node v. If a node has a negative margin, it indicates misclassification. Intuitively, *a larger margin suggests that a node is more resilient to attacks, whereas a smaller but positive margin suggests a higher potential for a successful attack.*

With this intuition in mind, we propose the introduction of a **cost-aware loss (CA-loss)** that takes into account the margins of nodes. By doing so, our aim is to assign higher priority to nodes with smaller but positive margins.

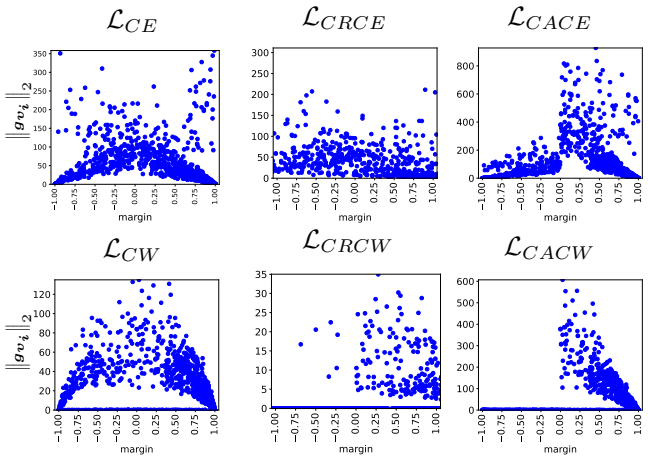

Fig. 1: Scatterplot of the nodes' margins with l_2 norm of their partial gradient matrices of cora dataset. The terms $\mathcal{L}_{CE}$, $\mathcal{L}_{CRCE}$ [13], and $\mathcal{L}_{CACE}$ respectively represent the negative log likelihood loss, the CR framework, and the CA framework and terms $\mathcal{L}_{CW}$, $\mathcal{L}_{CRCW}$, and $\mathcal{L}_{CACW}$ respectively represent the CW loss, the CR framework, and the CA framework

While a simplistic strategy is to rank the nodes based on margins and perturb them sequentially, this approach is computationally demanding and potentially suboptimal due to the complex interplay of nodes and edges in predictions. Instead, we refine the attack loss ℓ_{atk} by incorporating node margins to weight the nodes dynamically. The resulting CA-loss is defined as follows:

$$\mathcal{L}_{CA}\left(f_{\theta^*}, \widehat{G}\right) = \sum_{v \in V_U} w\left(v\right) \cdot \ell\left(f_{\theta^*}\left(\widehat{G}\right)_v, y_v\right), \tag{5}$$

where $w\left(v\right)$ is the weight of the node v. When all nodes are assigned equal weight, our CA-loss reduces to the conventional loss. To introduce the weight $w(v)$, which captures the inverse relationship between the node's margin $\varphi(v)$ and the weight, we utilize the exponential function. The weight $w(v)$ is defined as follows:

$$w(v) = \alpha \times e^{-\beta \times \varphi(v)^2}, \tag{6}$$

where α and β are tunable hyper-parameters. The weight $w(v)$ exponentially decreases as the node's margin $\varphi(v)$ increases. Figure 2 is a visualization of $w(v)$. This property ensures that the majority of perturbed edges are utilized to disrupt nodes with smaller margins during the attack process.

4. EXPERIMENTS

In this section, We present the experimental results by answering the following questions:

- $RQ1$:Can our CA attack loss framework enhance the efficacy of current poisoning attacks?

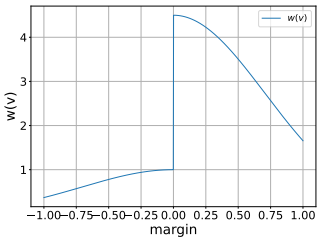

Fig. 2: Weights of nodes in Cora dataset with different margins.

Table 1: Dataset statistics.Following [1], we only consider the largest connected component (LCC).

Dataset	Nodes	Edges	Classes	Features
Cora	2,485	5,069	7	1,433
Citeseer	2,110	3,668	6	3,703
Polblogs	1,222	16,714	2	/

- $RQ2$:Can our CA attack loss framework serve as a universal framework for conventional poisoning attack losses?
- $RQ3$:Is our framework more computationally efficient compared to baselines?
- $RQ4$: How does our framework perform in terms of transfer learning?

4.1. Set Up

Datasets and GNN models. We evaluate our attacks on three benchmark graph datasets, i.e., Cora [16], Citeseer [17] and Polblogs [18]. Table 1 shows the basic statistics of these datasets. The datasets are partitioned into 10% labeled nodes and 90% unlabeled nodes. The true labels of the unlabeled nodes are hidden from both the attacker and the surrogate, serving only as a benchmark for assessing the effectiveness of the adversarial attacks. We employ the Graph Convolutional Network (GCN) [2] as our target GNN model.

Baseline Attacks. We choose DICE [19][1], MetaSelf [12][2] and Certify robustness(CR) inspired attack framework [13] as the base attack methods. Details of baselines are discussed in Section 1. For clarity in presentation, we denote the variant of MetaSelf utilizing the negative log likelihood loss as CE-MetaSelf. When employing the CW loss [14], it is referred to as CW-MetaSelf. When integrating the CR-inspired attack framework with MetaSelf using the negative log likelihood loss, we term it CR-CE-MetaSelf. Similarly, its adaptation with the CW loss is named CR-CW-MetaSelf.

[1] https://github.com/DSE-MSU/DeepRobust
[2] https://github.com/DSE-MSU/DeepRobust

Table 2: Hyperparameters of CA.

Dataset	$\alpha 1$	$\alpha 2$	$\beta 1$	$\beta 2$
Cora	4.5	1.0	1.0	1.0
Citeseer	4.5	1.0	1.0	1.0
Polblogs	1.0	1.0	0.5	0.1

Implementation Details All attacks are implemented in PyTorch and run on a Linux server with 16 core 1.0 GHz CPU, 24 GB RAM, and 10 Nvidia-RTX 3090Ti GPUs. Each experiment is repeated for ten times under different initial random seeds, and the uncertainties are presented by 95% confidence intervals around the mean in our tables. We separately set the perturbation budget Δ as 5% and 10% of the total number of edges in a graph. The specific parameters associated with our cost aware loss were selected by grid search to achieve optimal attack performance and details are in Table 2. For $\varphi(v) > 0$, we use $\alpha 1$ and $\beta 1$. Conversely, for $\varphi(v) < 0$, we employ $\alpha 2$ and $\beta 2$. For the baseline CR inspired attack framework, we adopted the parameter settings as outlined in their original paper and computed the certify robustness size at every $20th$ iteration during the generation of the poisoned graph.

4.2. Results and Analysis

Performance Comparison($RQ1$) Table 3 presents a comparison of our attack performance against other established methods. We can observe that our CA attack loss framework combined with negative log likelihood loss can enhance the CE-MetaSelf performance in all datasets. For instance, when attacking GCN with a 5% perturbation ratio, our CA-CE-MetaSelf demonstrates a gain of 9.58% and 9.74% over CE-MetaSelf on Cora and Citeseer, respectively. Moreover, CA-CE-Metaself has a relative improvement of 12.20% and 3.71% compared to CR-CE-MetaSelf, with a perturbation ratio of 10% on Cora and Citeseer, respectively. These findings indicate that our CA attack loss framework enhances poisoning attack performance more effectively within the same budget constraints.

Universality Analysis($RQ2$) Table 3 highlights the effectiveness of our weight assignment strategy when applied to the CW loss, underscoring the versatility of our approach. For example, when considering a 5% perturbation ratio, our proposed CA-CW-MetaSelf approach exhibits a relative gain of 3.58% and 3.06% over the CW-MetaSelf method for the Citeseer and Cora datasets, respectively. Additionaly, CW-CE-MetaSelf has a gain of 1.71% and 3.97% over the CR-CW-MetaSelf for the Cora and Citeseer datasets, respectively. Since we use the same parameters with negative log likelihood loss for CW loss to reduce the time for parameter adjustments, it may not be optimal for the CW loss which could result in suboptimal results compared to the baselines. The results of our attack can be improved with more detailed parameter tuning for the CW loss within the CA attack loss framework.

Computational Efficiency($RQ3$) We compared the computational efficiency of our method with the baselines, shown in Table 5. DICE has the highest computational efficiency since it depends on randomness without gradient derivation. The computational efficiency of CR-CE-MetaSelf and CR-CW-MetaSelf are the lowest because they need much time to compute certify robustness size for every node. CA-CE-MetaSelf and CA-CW-MetaSelf have similar computational efficiency with CE-MetaSelf and CW-MetaSelf. Therefore, the computational efficiency of CA attack loss framework is satisfactory while the performance is competitive.

Transfer Learning Performance Comparison($RQ4$) Table 4 shows the experimental results for the case where the structures of the victim model and the surrogate model are different. Both CA-CE-MetaSelf and CA-CW-MetaSelf have competitive transfer performance compared to other baselines. In evaluations using GATs on Cora, our CA-CE-MetaSelf outperformed CR-CE-MetaSelf by 2.97%. Similarly, with GraphSAGE on Citeseer, CA-CW-MetaSelf surpassed CR-CW-MetaSelf by 3.09%.

5. CONCLUSION

We investigate graph poisoning attacks on graph neural networks and present an innovative Cost Aware Poisoning Attack Loss Framework(CA-attack) utilizing the concept of classification margin. Our research begins by examining the budget inefficiencies present in previous attack methods. Subsequently, we develop a cost-aware loss framework that assigns dynamic weights to victim nodes based on their margins. Through evaluations conducted on various datasets, we demonstrate that our attack loss can considerably decrease the budget waste and improve the performance of existing attacks.

Table 3: Experimental results of attacks for GCNs. The best results from each experiment are bold.

Dataset	Cora		Citeseer		Polblogs	
Pert Rate(%)	5%	10%	5%	10%	5%	10%
Clean	84.61±0.28		73.95±0.28		95.34±0.19	
DICE	82.79±0.17	81.92±0.22	72.32±0.24	71.59±0.28	86.54±0.42	80.71±0.29
CE-MetaSelf	76.56±0.27	66.96±0.55	71.49±0.39	66.20±0.30	77.22±0.34	69.73±0.44
CR-CE-MetaSelf	75.79±0.34	68.50±0.29	65.28±0.43	56.93±0.33	78.68±0.32	70.43±0.85
CA-CE-MetaSelf	**66.98±0.56**	**56.30±0.42**	**62.58±0.28**	**53.22±0.30**	**76.58 ±0.46**	**69.43±0.22**
CW-MetaSelf	72.75±0.50	61.85±1.18	64.37±1.07	**52.17±0.25**	83.71±0.66	79.45±0.47
CR-CW-MetaSelf	71.40±0.61	60.84±0.45	64.76±0.88	54.98±1.29	**83.09±0.92**	79.87±1.02
CA-CW-MetaSelf	**69.69±0.64**	**58.50±0.73**	**60.79±0.51**	53.28±0.62	83.24±0.45	**76.99±0.46**

Table 4: Experimental results of transfer learning on GAT and GraphSage with 5% perturbation rate.

Dataset	Cora		Citeseer		Polblogs	
Victim Model	GAT	GraphSage	GAT	GraphSage	GAT	GraphSage
Clean	83.54±1.41	83.21±1.54	74.11±1.69	73.28±1.02	94.19±1.77	93.73±1.98
DICE	82.62±2.0	81.51±1.33	73.75±2.25	72.61±1.58	87.63±3.01	83.81±2.98
CEMetaSelf	80.24±3.58	77.93±1.53	72.80±2.57	71.31±1.41	87.26±2.26	83.26±3.36
CR-CEMetaSelf	79.62±4.50	76.59±1.33	69.43±3.57	66.41±1.39	88.34±2.24	84.72±2.55
CA-CEMetaSelf	**76.65+-2.66**	**72.55+-2.88**	**68.75+-3.24**	**65.23+-2.95**	**86.48+-1.77**	**82.11+-3.31**
CWMetaSelf	78.06±5.77	74.20±1.82	68.89±3.24	65.40±1.64	89.82±3.10	86.96±3.73
CR-CWMetaSelf	78.44±3.21	74.95±1.54	69.16±3.52	66.00±1.73	**90.34±3.52**	87.71±3.48
CA-CWMetaSelf	**76.38+-3.56**	**71.69+-2.17**	**66.72+-4.58**	**62.14+-1.84**	90.88+-3.35	**87.45+-2.54**

Table 5: Comparison of computational efficiency.

Methods	Cora	Citeseer	Polblogs
DICE	0.04s	0.01s	0.06s
CE-MetaSelf	107.46s	97.90s	114.07s
CW-MetaSelf	158.39s	129.12s	236.52s
CR-CE-MetaSelf	84533.67s	53089.36s	42780.71s
CR-CW-MetaSelf	87566.54s	56345.78s	53390.56s
CA-CW-MetaSelf	153.58s	122.74s	200.68s
CA-CE-MetaSelf	104.39s	107.88s	112.37s

6. REFERENCES

[1] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang, "Graph structure learning for robust graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[2] Thomas N. Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[3] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio, "Graph attention networks," in *Proceedings of the 5th International Conference on Learning Representations*, 2018.

[4] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin, "Graph neural networks for social recommendation," in *The World Wide Web Conference*, 2019.

[5] Anshika Chaudhary, Himangi Mittal, and Anuja Arora, "Anomaly detection using graph neural networks," in *International Conference on Machine Learning, Big Data, Cloud and Parallel Computing*, 2019.

[6] Jianheng Tang, Jiajin Li, Ziqi Gao, and Jia Li, "Rethinking graph neural networks for anomaly detection," in *Proceedings of the 39th International Conference on Machine Learning*, 2022.

[7] Jingcan Duan, Siwei Wang, Pei Zhang, En Zhu, Jingtao Hu, Hu Jin, Yue Liu, and Zhibin Dong, "Graph anomaly detection via multi-scale contrastive learning networks with augmented view," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[8] Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, and Bo Long, "Graph neural networks for natural language processing: A survey," *Foundations and Trends in Machine Learning*, vol. 16, 2023.

[9] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song, "Adversarial attack on graph structured data," in *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[10] Binghui Wang and Neil Zhenqiang Gong, "Attacking graph-based classification via manipulating the graph structure," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[11] Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin, "Topology attack and defense for graph neural networks: An optimization perspective," 08 2019, pp. 3961–3967.

[12] Daniel Zügner and Stephan Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *Proceedings of the 6th International Conference on Learning Representations*, 2019.

[13] Binghui Wang, Meng Pang, and Yun Dong, "Turning strengths into weaknesses: A certified robustness inspired attack framework against graph neural networks," *Proceedings of the 22th IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[14] Nicholas Carlini and David Wagner, "Towards evaluating the robustness of neural networks," in *ieee symposium on security and privacy (sp)*, 2017.

[15] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann, "Adversarial attacks on neural networks for graph data," in *ACM Special Interest Group on Knowledge Discovery and Data Mining*, 2018.

[16] Andrew McCallum, Kamal Nigam, Jason D. M. Rennie, and Kristie Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, 2000.

[17] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, 2008.

[18] Lada Adamic, "The political blogosphere and the 2004 u.s. election: Divided they blog," *Proceedings of the 3rd International Workshop on Link Discovery*, 2005.

[19] Marcin Waniek, Tomasz Michalak, Talal Rahwan, and Michael Wooldridge, "Hiding individuals and communities in a social network," *Nature Human Behaviour*, 2018.