Dynamic Gradient Influencing for Viral Marketing Using Graph Neural Networks

Saurabh Sharma
University of California, Santa Barbara
saurabhsharma@ucsb.edu

Ambuj Singh
University of California, Santa Barbara
ambuj@ucsb.edu

Abstract

The problem of maximizing the adoption of a product through the process of viral marketing in social networks is of extreme importance. Accordingly, we use Graph Neural Networks (GNN) to model the adoption of products in a data-driven way using both topological and attribute information. We propose the novel Dynamic Viral Marketing (DVM) problem of finding the minimum budget and minimal set of dynamic topological and attribute changes to attain a spread goal. We show that DVM is NP-Hard and is connected to influence maximization (IM). Motivated by this connection, we develop the Dynamic Gradient Influencing (DGI) model, which uses gradient-guided node flipping to search for the optimal perturbations and targets low-budget and high influence non-adopters in discrete steps. We use an efficient strategy for computing node budgets and develop the Meta-Influence heuristic for assessing a node's downstream influence. We evaluate our proposed DGI against multiple non-gradient and gradient baselines to show the efficacy of our approach on real-world attributed networks. Experiments reveal that DGI discovers realistic cascade patterns through intermediary low-degree nodes that are confidently classified as adopters by the GNN.

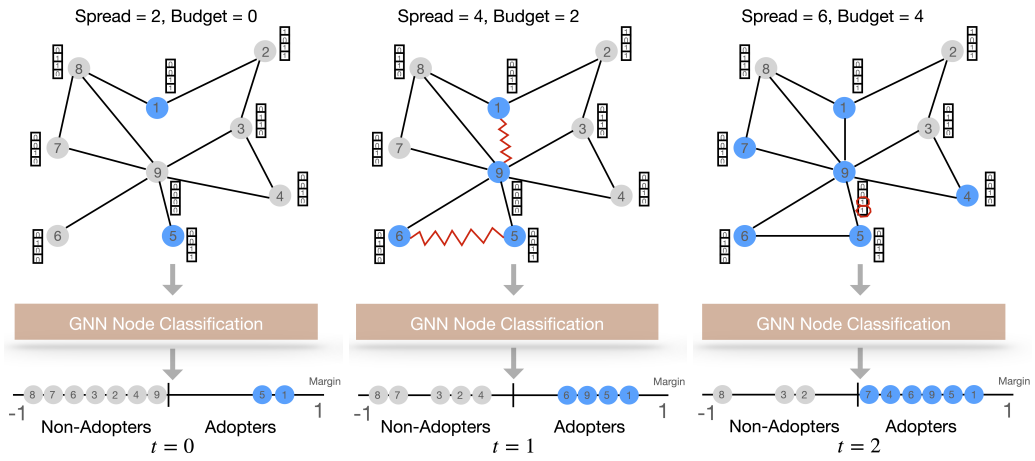
1 Introduction

Viral marketing is a highly significant strategy used to maximize the spread of adoption of products [1, 2], which relies upon diffusion within a social network of users that is similar to the spread of infection [3]. However, prior work [4] concentrates on linearized propagation models of viral phenomena which focus on static topologies and *ignore rich node attribute information*. Further, while finding the influential seed set has been extensively studied, the problem of making *dynamic topological and attribute perturbations* to maximize spread hasn't been addressed.

Therefore, to use attribute information, we use non-linear Graph Neural Networks (GNNs) [5, 6] to learn a propagation model directly from attributed network data and then use it to forecast future states of the spread after the network is perturbed. For dynamic perturbations, we consider a realistic model that can be used to strategically accelerate spread from adopters to non-adopters. Accordingly, at any given time, (1) new edges can be added only between adopters and non-adopters, as in referral marketing [7], and (2) adopters can further adopt similar products or products from partnering firms by flipping corresponding attributes from 0 to 1, as in joint or co-marketing [8].

Consequently, we propose the Dynamic Viral Marketing (DVM) problem, which seeks to find the minimum budget and minimal dynamic perturbation set to attain a spread goal. We show that DVM is a NP-Hard problem and is connected to the influence maximization (IM) problem [4]. Motivated by this connection, we develop the Dynamic Gradient Influencing (DGI) model, which uses gradient-guided node flipping to search for the optimal perturbations and targets low-budget and high influence non-adopters in discrete steps. We develop an efficient node flipping budget computation approach using bisection search to maintain node budgets at each step. To estimate a node's downstream influence, we develop the gradient based Meta Influence heuristic and corresponding Meta Attribute Flips to increase the potency of outgoing edge perturbations.

S. Sharma et al., Dynamic Gradient Influencing for Viral Marketing Using Graph Neural Networks (Extended Abstract). Presented at the Third Learning on Graphs Conference (LoG 2024), Virtual Event, November 26–29, 2024.



(a) Initial State **(b)** Referral induced flip **(c)** Co-marketing induced flip

Figure 1: Overview of the Dynamic Viral Marketing (DVM) problem.

2 Dynamic Viral Marketing Problem

Formally, consider $G^t = (A^t, X^t, Y^t)$, a series of undirected and unweighted dynamic attributed networks, observed at discrete time steps $t = 1, \ldots, T$, where A^t represents the adjacency matrix containing node connectivity, X^t represents a binary node feature matrix containing adoption labels for related products, and Y^t represents the binary adoption labels for the target product. We assume that the adoption labels are given by $Y^t = f(A^t, X^t, Y^0)$, where $f()$ is a general propagation model which diffuses the initial labels Y^0 using the network structure and attributes at time t. The sets of adopters and non-adopters at time t are denoted by S^t and D^t, and the resulting spread by $\sigma(G^t)$,

$$S^t = \sum_v \mathbb{1}[y_v^t = 1], \quad D^t = \sum_v \mathbb{1}[y_v^t = 0], \quad \sigma(G^T) = |S^T| \tag{1}$$

Due to the representation learning ability of graph neural networks [5, 6, 9, 10] through the propagation of feature and label information, we use a GNN as our propagation model $f()$. Specifically, a GNN f_θ is trained on the initial network G^0 and its parameters θ are then fixed. Thereafter, we use *self-labeling* at time t: the predictions from the GNN on the network G^t yield the adoption states Y^t. Therefore, both our seed nodes, Y^0, and our propagation model, $f()$, are data-driven.

The dynamic transitions of the network from G_{t-1} to G_t are constrained as follows:

- *Referral marketing*: Edge insertions can be made only between S^{t-1} and D^{t-1} at time $t-1$ (adopters make connections to non-adopters). The total cost for structural changes is $|A^t - A^{t-1}|$.
- *Co-marketing*: Features of nodes in S^{t-1} can flip from 0 to 1 at time $t-1$ (adopters adopt similar products to the target product). The total cost for attribute changes is $|X^t - X^{t-1}|$.

As all the changes are made incrementally, the total cost incurred is given by $|A^T - A^0| + |X^T - X^0|$.

Dynamic Viral Marketing (DVM) We now state the DVM optimization problem of finding the minimum budget, $\mu()$, and minimal set of changes to reach a spread ϕ,

$$\underset{A^1,\ldots,A^T,X^1,\ldots,X^T}{\text{arg-min}} \quad \sigma(G^T) \geq \phi, \quad \mu(\phi, G_0) = |A^T - A^0| + |X^T - X^0| \tag{2}$$

Note that while the total budget is a function of the final adjacency and feature matrices, to solve DVM a *sequence* of structural and attribute changes under the referral and co-marketing contraints are required. Further, while we use a uniform cost model, the problem can easily be extended to bespoke settings by using edge-specific and attribute-specific costs. Figure 1 gives an overview of the DVM problem. We prove its NP-Hardness in the Appendix. Moreover, in Thm 1 in the Appendix, we draw a connection with influence maximization (IM) that shows node flipping in DVM occurs when the sum of the dynamic influence edge weights exceeds the dynamic node threshold.

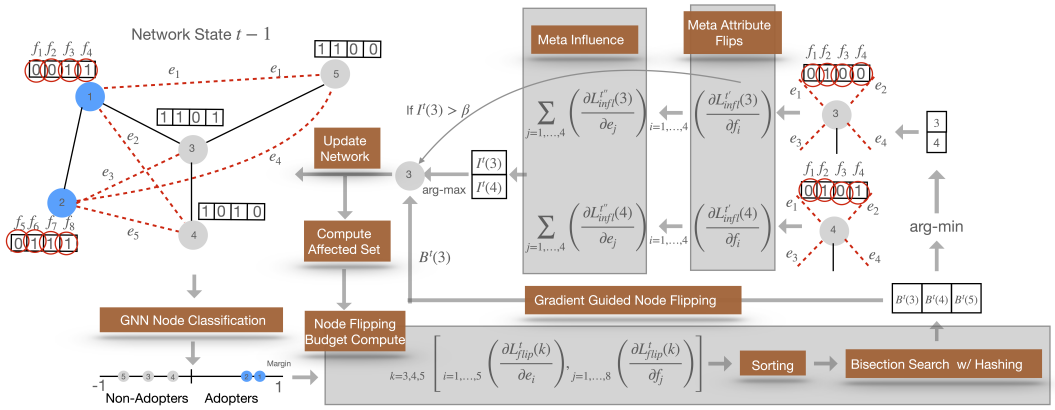


Figure 2: Overview of the DGI model. Red lines/circles indicate candidate perturbations.

3 Dynamic Gradient Influencing Model

Motivated by the criterion for node flipping in Thm. 1, we develop the Dynamic Gradient Influencing (DGI) model to solve the DVM problem. Figure 2 provides an overview of the DGI model. DGI uses Gradient-Guided Node Flipping, to flip a particular candidate node in each step. At each step t, the candidate node to flip, v^t, is given by,

$$\underset{v \in N}{\text{arg-max}}\, I^t(v), \text{ where } N = \underset{w \in D}{\text{arg-min}}\, B^t(w) \tag{3}$$

where $B^t(w)$ denotes the budget to flip node w, and $I^t(v)$ denotes the Meta Influence of node v.

3.1 Gradient-Guided Node Flipping

In DGI, the core functionality for flipping nodes is accomplished through gradients on the restricted set of perturbations arising from the referral and co-marketing constraints of DVM. We consider the negative cross-entropy loss as our flip loss for the chosen candidate node $v \in D$. We gather gradients from the flip loss and formulate the gradient ranking scheme for the restricted set of perturbations at each step. More details on this formulation can be found in the Appendix.

3.2 Node Flipping Budget Computation

To compute the minimum budget $B^t(v)$ that converts node v, we use the bisection method [11, 12]. For each node, we compute the sorted gradients once and run bisection search over these gradients. Since budgets need to be recomputed for all non-adopters at every step, we make the algorithm faster by hashing node budgets and only recomputing budgets for nodes whose budget has changed. The recompute set R^t is defined as the set of nodes whose logit scores changed in the previous time step. More details on the budget compute algorithm can be found in the Appendix.

3.3 Meta Influence using Meta Attribute Flips

To characterize nodes that have high influence so that adoptions can sequentially cascade, we develop the Meta Influence heuristic to model long-range effects and estimate downstream influence. Firstly, we make Meta Attribute Flips by restricting feature perturbations to the features of node v. We consider an influence loss which is the sum of a CW-type loss [13] on all non-adopters nodes. The Meta Attribute Flips are the top-k ranked perturbations from the gradients on the influence loss. For finding Meta Influence, we restrict the outgoing edge perturbations from node v to the non-adopters D. Thereafter, conditioned on Meta Attribute Flips, we compute gradients on the edge perturbations from the influence loss. Finally, the Meta Influence $I^t(v)$ of a node v is defined as the normalized gradient score of the node's edge perturbations averaged over the number of non-adopters.

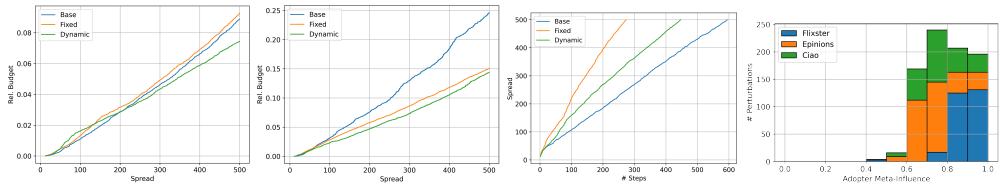
During candidate selection, Meta Influence is used for tiebreaking between equal budget nodes (Eq. 3). Further, we threshold on Meta Influence to perform Meta Attribute Flips at candidate nodes after flipping. Using Meta Influence, we estimate which nodes will have high influence on their outgoing edges after Meta Attribute Flips, and judiciously allocate budget for Meta Attribute Flips. More details on the exact form of Meta Influence and the DGI algorithm can be found in the Appendix.

Table 1: Comparison of DGI variants to baselines. Numbers indicate μ, the minimum budget to spread to 500 nodes, and AUC of budget-spread curves. GCN and SAGE are used as the GNN propagation backbones. Dynamic DGI consistently achieves the minimum budget and AUC.

	Flixster				Epinions				Ciao			
	GCN		SAGE		GCN		SAGE		GCN		SAGE	
	μ	AUC	μ	AUC	μ	AUC	μ	AUC	μ	AUC	μ	AUC
Degree	2551	19.48	6573	150.50	22076	215.33	21125	328.15	25162	349.48	45291	631.66
Margin	8136	173.70	7655	177.71	26109	366.43	18550	284.77	20814	287.11	41077	595.69
GradArgmax	852	21.62	691	22.25	2729	50.28	1003	22.54	6091	91.38	1100	23.67
MiBTack	843	21.78	583	16.77	2828	50.70	866	21.03	5111	80.95	1035	21.91
Base	797	19.44	506	10.54	3235	54.01	831	15.44	4231	66.02	878	14.66
Fixed	831	20.78	714	15.79	1985	36.41	1297	23.75	2221	36.20	1094	19.82
Dynamic	**667**	**18.27**	**494**	**10.17**	**1893**	**31.93**	**803**	**15.18**	**2096**	**33.84**	**821**	**13.33**

4 Experiments

Evaluation. We utilize real-world attributed datasets [14, 15] for evaluation. We consider 3 variants of DGI: (1) Base: DGI without Meta Attribute Flips, (2) Fixed: DGI with fixed Meta Attribute Flips, (3) Dynamic: DGI with optimally chosen threshold for Meta Attribute Flips. The efficacy of the DGI variants and baselines is evaluated using the minimum budget for spread at $\phi = 500$ target nodes. We also report the Area Under Curve (AUC) of the budget-spread curve. For the purpose of calculating AUC we normalize the budget by the sum of the number of edges and turned on features. We compare DGI variants to the baselines using budget and AUC at $\phi = 500$ in Table 1. DGI variants outperform the baselines in all the scenarios. Among the variants, Dynamic does the best by judiciously applying Meta Attribute Flips through thresholding on Meta Influence, followed by Base and then Fixed which overspends on Meta Attribute Flips. Further, we plot the budget spread curves on Flixster and Epinions in Fig. 4. Dynamic DGI consistently achieves the minimum budget across all spread levels. To understand the time evolution of the spread for different variants, we plot spread as a function of time in Fig. 3c. Due to the accelerating effect of Meta Attribute Flips, Fixed spreads fastest, followed by the economical Dynamic and the conservative Base approach. Moreover, to validate the effectiveness of Meta Influence and Meta Attribute Flips, we plot the histogram of perturbations contributed by nodes with respect to their Meta Influence in Figure. 3d. We observe the high correlation of Meta Influence to the number of contributed perturbations across all datasets. More details and experiments can be found in the Appendix.



(a) **(b)** **(c)** **(d)**

Figure 3: (a)-(b) Budget spent as a function of increasing spread with GCN as the GNN propagation model on Flixster and Epinions respectively. (c) Spread achieved with increasing number of time steps. (d) Histogram of perturbations contributed by intermediary adopter nodes with increasing Meta Influence, where scaling is used for mapping Meta Influence to [0,1].

5 Conclusion and Future Work

We proposed the novel Dynamic Viral Marketing (DVM) problem to find the minimum budget and minimal dynamic perturbation set to attain a spread goal, where the propagation model is a non-linear GNN and perturbations are restricted by referral and co-marketing constraints. We developed the Dynamic Gradient Influencing (DGI) model, which targets non-adopters with low budget and high influence. DGI uses gradient-guided node flipping to order perturbations and consists of an efficent budget computation approach, a novel Meta Influence heuristic and Meta Attribute Flips to increase a node's influence. We comprehensively evaluated DGI on 3 real world attributed networks and showed that it outperforms multiple gradient and non-gradient baselines. We leave the question of competitive spreading in the DVM framework as future work.

References

[1] Pedro M. Domingos and Matthew Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001. 1, 6

[2] Matthew Richardson and Pedro M. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002. 1, 6

[3] Stephen Morris. Contagion. *The Review of Economic Studies*, 67(1), 2000. 1, 6

[4] David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. *Theory Comput.*, 2015. 1, 6, 8

[5] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016. 1, 2, 6, 7, 12

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 2016. 1, 2

[7] Francis A Buttle. Word of mouth: understanding and managing referral marketing. *Journal of strategic marketing*, 6(3), 1998. 1

[8] Ironclad Journal. Joint marketing agreements. https://ironcladapp.com/journal/contracts/joint-marketing-agreement/, 2024. Accessed: 2024-08-29. 1

[9] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *7th International Conference on Learning Representations, ICLR*, 2019. 2

[10] Hongwei Wang and Jure Leskovec. Unifying graph convolutional neural networks and label propagation. 2020. 2

[11] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 3, 9

[12] Pedro Tabacof and Eduardo Valle. Exploring the space of adversarial images. In *2016 international joint conference on neural networks (IJCNN)*. IEEE, 2016. 3, 9

[13] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*. Ieee, 2017. 3, 10

[14] Jiliang Tang, Huiji Gao, and Huan Liu. mtrust: discerning multi-faceted trust in a connected world. In *Proceedings of the Fifth International Conference on Web Search and Web Data Mining, WSDM*, 2012. 4, 12

[15] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys*, 2010. 4, 12

[16] Jon Kleinberg. *Networks, Crowds, and Markets*. Cambridge University Press, 2010. 6

[17] Mark Granovetter. Threshold models of collective behavior. *American journal of sociology*, 1978. 6, 8

[18] Mark Newman. *Networks*. Oxford university press, 2018. 6

[19] Amit Goyal, Francesco Bonchi, and Laks V. S. Lakshmanan. A data-based approach to social influence maximization. *Proc. VLDB Endow.*, 5(1):73–84, 2011. 6

[20] Ying Wang and Yanhao Wang. Opinion-aware influence maximization in online social networks. In *6th International Conference on Data Science and Information Technology, DSIT*, 2023. 6

[21] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 2017. 6, 7, 12

[22] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations*, 2018. 6

[23] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning,*, 2019. 6

[24] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 2021. 6

[25] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International conference on machine learning*, 2018. 7, 9, 12

[26] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations (ICLR)*, 2019.

[27] Kaidi Xu, 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. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019. 7, 9, 12

[28] Mengmei Zhang, Xiao Wang, Chuan Shi, Lingjuan Lyu, Tianchi Yang, and Junping Du. Minimum topology attacks for graph neural networks. In *Proceedings of the ACM Web Conference 2023*, 2023. 7, 12

[29] Sahil Manchanda, Akash Mittal, Anuj Dhawan, Sourav Medya, Sayan Ranu, and Ambuj K. Singh. GCOMB: learning budget-constrained combinatorial algorithms over billion-sized graphs. In *Advances in Neural Information Processing Systems*, 2020. 7

[30] Sanjay Kumar, Abhishek Mallik, Anavi Khetarpal, and Bhawani Sankar Panda. Influence maximization in social networks using graph embedding and graph neural network. *Inf. Sci.*, 607:1617–1636, 2022. 7

[31] Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of graph neural networks at scale. *Advances in Neural Information Processing Systems*, 34, 2021. 9

[32] Felix Mujkanovic, Simon Geisler, Stephan Günnemann, and Aleksandar Bojchevski. Are defenses for graph neural networks robust? *Advances in Neural Information Processing Systems*, 35, 2022. 9

[33] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI*, 2020. 12

[34] Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*, 2020. 12

A Related Work

Models of Spreading Network Processes. The modeling of the diffusion of innovations in a network through the process of social contagion is a long studied topic [16]. [17] developed a threshold based model of collective behaviour where individuals are influenced by the proportion of others who come to a particular decision. [3] studied a coordination game of direct benefits from aligning choices with neighbors in a social network. In epidemiology, the spread of biological disease is studied using probabilistic transmission models of Susceptible, Infected and Recovered (SIR) individuals [18]. The use of the network value of customers for marketing is first explored in [1, 2]. Consequently, Influence Maximization (IM), the problem of finding the most influential seed set for viral marketing has been studied extensively [4, 19, 20]. Dynamic Viral Marketing lies within the broad class of spreading network processes and we show how its connected to IM.

Graph Neural Networks. Graph Neural Networks (GNN) are message passing neural networks that operate on attributed networks and have shown great success in problems such as node classification, link prediction, recommendation systems, and community detection. Various GNN architectures have been proposed since their first inception- Graph Convolutional Networks [5], GraphSAGE [21], Graph Attention Networks [22], Simplifying Graph Convolutional Networks [23]. We refer the reader to [24] for an extensive survey of graph neural networks. We use GNNs as the underlying propagation model for Dynamic Viral Marketing.

Gradient-based Network Optimization. Gradient-based network optimization is used in many combinatorial network optimization problems. In the context of adversarial attacks on GNNs, perturbations are made using gradient optimization on the input network strucure to bring down the

accuracy of a GNN classifier [25–27]. Global attacks with dynamic budget adjustment for Topology PGD [27] are considered by [28]. In [29], reinforcement learning policies are optimized to solve Maximum Coverage, Vertex Cover, and Influence Maximization problems on networks. In [30], graph neural networks are optimized to create graph embeddings and predict influence of nodes for solving Influence Maximization. We use gradient-based network optimization to develop Dynamic Gradient Influencing.

B Notation and Preliminaries

Consider a graph $G = (A, X)$, with the associated adjacency matrix $A \in \{0,1\}^{n\times n}$ and node attribute matrix $X \in \{0,1\}^{n\times d}$ respectively, and node labels $Y \in \{0,1\}^n$. We refer to the associated node-ids as $\mathcal{V} = \{1, \ldots, n\}$. We denote the node feature $x_v \in \{0,1\}^d$, and the node label $y_v \in \{0,1\}$. The set of adopters and non-adopters is denoted as S and D respectively. For convenience, we denote the sub-matrix of A containing node connectivity from a set of nodes S to a set of nodes D as $A_{S,D}$, the sub-matrix of X containing features for a set of nodes S as X_S, the vectors of ones and zeros as $\mathbf{1}$ and $\mathbf{0}$ respectively. We denote the weights on edge and feature perturbations as P_A and P_X respectively. The gradient scores on edge and feature perturbations are denoted by $\hat{P}_A$ and $\hat{P}_X$ respectively. For a dynamically changing network, the superscript t is used to indicate the variable at time t; we drop the superscript if it is clear from the context.

We consider the large family of graph neural networks [5, 21] to construct layerwise hidden representations and finally output classifier logit scores $Z \in \mathbb{R}^{n\times 2}$. For an L-layer GNN,

$$H_l = \sigma(\hat{A}W_l H_{l-1}), \qquad H_0 = X, \qquad Z = H_L$$

where W_l refers to the learnable GNN parameters at layer l, σ is a nonlinear activation function, and $\hat{A}$ is the GNN propagation matrix. For Graph Convolutional Networks (GCN) [5], $\hat{A} = \bar{\Delta}^{-\frac{1}{2}}\bar{A}\bar{\Delta}^{-\frac{1}{2}}$, where $\bar{A} = A + I$ and $\bar{\Delta}$ is the associated degree matrix. For GraphSAGE [21] with mean pooling, $\hat{A} = \bar{\Delta}^{-1}\bar{A}$. The predicted labels $y'_v \in \{0,1\}$ for each node $v \in \mathcal{V}$ are given by the class with the maximum logit score. In the typical semi-supervised learning scheme for node classification, referred to as transductive learning, the GNN parameters W are learned by minimizing the cross-entropy classification loss,

$$\mathcal{L}_{tr}(v) = -\log \sigma(z_v)_{y_v} \tag{4}$$

where z_v denotes the logit scores for node v and σ is the softmax function.

C NP-Hardness of DVM

Consider an instance of the NP-Complete *Knapsack* problem, defined by a maximum value V and a maximum weight W, and a set of n items $X = \{(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)\}$ where v_i and w_i denote the i^{th} item's value and weight respectively. We wish to know whether there exists a subset of items $Z \subset X$ with total weight $\sum_{i\in Z} w_i \leq W$ and total value $\sum_{i\in Z} v_i \geq V$. We show that this can be viewed as a special case of the Dynamic Viral Marketing problem.

Given an arbitrary instance of the Knapsack problem, consider a weighted star network with target node t at the center which is connected to n source nodes. Each node has a single feature, whose value at node t is 0 and node i is $x_i = 1 - w_i$. The weight on the edge between node i and t is set to $a_i = v_i$. The initial label on node t is 0 and the initial label on the other nodes is 1. The cost of changing the feature at a node from a value of $1 - w_i$ to 1 is w_i. The GNN classifier parameters are tuned such that the prediction on node t flips from 0 to 1 when the weighted sum of its neighborhood of n nodes with feature value $x_i = 1$, is at least V. In other words, the Knapsack problem is solvable if $\sum_{i\in[n]} a_i \mathbb{1}[x_i = 1] \geq V$ and $\sum_{i\in[n]} w_i \mathbb{1}[x_i = 1] \leq W$. Thus, if the DVM problem can find feature flips costing at most W at t's neighbors whose edge weights sum to at least V then the Knapsack must be solvable. □

D Connection of Influence Maximization to DVM

We draw an interesting connection between DVM and the related problem of influence maximization (IM) [4]. Consider the linear threshold propagation model [17], where nodes $i \in [n]$ randomly

choose a threshold $\theta_i \in [0, 1]$ and incoming influence edge weights $I_{i,j}$ such that $\forall i, \sum_j I_{i,j} \leq 1$. The propagation unfolds in discrete time steps, and if the set of active nodes at any given step is S, then an inactive node becomes active if the following constraint is satisfied,

$$\sum_{j \in S} I_{i,j} \geq \theta_i \tag{5}$$

While the objective in IM is to find a set of seed nodes for maximizing spread, we instead search for a sequence of dynamical changes to maximize spread. Despite the difference, given the set of adopters S^{t-1} at step $t-1$, the criterion for a node to flip in DVM has a similar form as Eq. 5. Suppose that the L-layer GNN f_θ has the associated L-step random walk propagation matrix M. Then the following theorem characterizes the dynamic node thresholds and influence edge weights,

Theorem 1. *Let the vector $\varepsilon_j^t = x_j^t - x_j^{t-1}$ denote the change in the feature of node j from time $t-1$ to t. Further, let the matrix $\xi = M^t - M^{t-1}$ denote the change in the L-step random walk matrix M from time $t-1$ to t. Then the dynamic threshold and influence edge weights for node i to flip at time t according to the criterion in Eq. 5 are given by,*

$$\theta_i^t = z_{i,0}^{t-1} - z_{i,1}^{t-1} \tag{6}$$

$$I_{i,j}^t = M_{i,j}^t \alpha^T \varepsilon_j^t + \xi_{i,j}^t \alpha^T x^{t-1} \tag{7}$$

where α is a vector which depends on the parameters θ of the GNN.

Proof. Consider a L-layer GNN without non-linearities. Then the criterion to flip node i at time t can be expressed in terms of the L-step random walk matrix M and the weights W_l,

$$\sum_{j \in N(i)} M_{ij}^t w_{L,1}^T \bar{W} x_j^t > \sum_{j \in N(i)} M_{ij}^t w_{L,0}^T \bar{W} x_j^t \tag{8}$$

where $w_{L,1}$ and $w_{L,0}$ represents the final layer classifier vectors for the two classes, and $\bar{W} = \prod_{i=1}^{L-1} W_l$ is the combined feature transform of the first $L-1$ layers.

The criterion can be equivalently written as,

$$\sum_{j \in N(i)} [M_{ij}^t w_{L,1}^T \bar{W}(x_j^t - x_j^{t-1}) + (M_{ij}^t - M_{ij}^{t-1}) w_{L,1}^T \bar{W} x_j^{t-1} + M_{ij}^{t-1} w_{L,1}^T \bar{W} x_j^{t-1}] >$$
$$\sum_{j \in N(i)} [M_{ij}^t w_{L,0}^T \bar{W}(x_j^t - x_j^{t-1}) + (M_{ij}^t - M_{ij}^{t-1}) w_{L,0}^T \bar{W} x_j^{t-1} + M_{ij}^{t-1} w_{L,0}^T \bar{W} x_j^{t-1}] \tag{9}$$

Rearranging terms,

$$\sum_{j \in N(i)} [M_{ij}^t (w_{L,1}^T - w_{L,0}^T) \bar{W}(x_j^t - x_j^{t-1}) + (M_{ij}^t - M_{ij}^{t-1})(w_{L,1}^T - w_{L,0}^T) \bar{W} x_j^{t-1}] >$$
$$\sum_{j \in N(i)} M_{ij}^{t-1} w_{L,0}^T \bar{W} x_j^{t-1} - \sum_{j \in N(i)} M_{ij}^{t-1} w_{L,1}^T \bar{W} x_j^{t-1} \tag{10}$$

Denoting $\alpha = \bar{W}^T (w_{L,1} - w_{L,0})$, and observing that the right hand side is the logit margin at time $t-1$, the above simplifies to,

$$\sum_{j \in N(i)} [M_{ij}^t \alpha^T \epsilon_j^t + \xi_{i,j}^t \alpha^T x^{t-1}] > z_{i,0}^{t-1} - z_{i,1}^{t-1} \tag{11}$$

□

Intuitively, the dynamic node threshold depends on its logit margin, and the dynamic influence edge weights depend on both the feature change ε_j and the random walk propagation change $\xi_{i,j}$. Theorem 1 suggests that budget should be spent on changes which contribute most to the incoming influence weights and push the node just beyond its threshold.

E More details on Gradient-Guided Node Flipping

The only changes that can happen to the adjacency matrix A and feature matrix X are restricted to the submatrices $A_{S,D}$ and X_S respectively. To rank the set of restricted perturbations, we define,

$$A^t_{S,D} = A^{t-1}_{S,D} + P^t_A \circ (\mathbf{1}\mathbf{1}^T - A^{t-1}_{S,D}),\ A^t_{D,S} = (A^t_{S,D})^T \tag{12}$$

$$X^t_S = X^{t-1}_S + P^t_X \circ (\mathbf{1}\mathbf{1}^T - X^{t-1}_S) \tag{13}$$

where P^t_A and P^t_X represent the weights on the edge and feature perturbations, $S = S^{t-1}$ and $D = D^{t-1}$, and $\circ$ denotes the Hadamard product. We initialize $P^t_A = \mathbf{0}\mathbf{0}^T$ and $P^t_X = \mathbf{0}\mathbf{0}^T$.

While it's an NP-Hard combinatorial optimization problem to find the minimal perturbation set that flip a node, given that the adjacency and feature matrices are both discrete, first-order gradients work well enough in practice to find the required perturbations [25, 27]. We consider the negative cross-entropy loss as our flip loss for the chosen candidate node $v \in D$,

$$L^t_{flip}(v) = \log \sigma(z^{t-1}_v)_0 \tag{14}$$

Note that $y_v = 0$ for non-adopters and $y_v = 1$ for adopters. We then compute non-negative gradient scores on edge perturbations, $\hat{P}^t_A$, and feature perturbations, $\hat{P}^t_X$, with respect to the flip loss,

$$\hat{P}^t_A = \max\left(\left[\frac{\partial L^t_{flip}(v)}{\partial P^t_A}\right], 0\right), \quad \hat{P}^t_X = \max\left(\left[\frac{\partial L^t_{flip}(v)}{\partial P^t_X}\right], 0\right) \tag{15}$$

Note that we only compute the gradients once for all the perturbations. While it's possible to recompute gradients after every perturbation [25], we find that this is not that necessary to find the minimal set of perturbations. Moreover, as shall be shown, by merging and sorting the perturbations using the gradients, we can find the minimal perturbation set and minimum budget efficiently. Finally, suppose the minimum budget required to convert v is $B(v)$, then we find the top-B(v) indices in the union of edge and feature perturbations,

$$\hat{P}^t = \text{sort}(\text{merge}(\hat{P}^t_A, \hat{P}^t_X)) \qquad i^t_A, i^t_X = \text{argtop-k}_{k=B(v)}\, \hat{P} \tag{16}$$

Using the index sets i^t_A and i^t_X we can make the updates to the network to convert v,

$$[A^t_{S,D}]_{i^t_A} \leftarrow 1, \qquad A^t_{D,S} \leftarrow (A^t_{S,D})^T \qquad [X^t_S]_{i^t_X} \leftarrow 1 \tag{17}$$

F More details on Node Flipping Budget Computation

The budget needed to flip a node v depends on both the logit margin in Eq. 6 and the node's degree $\deg(v)$ [31, 32]. In the adversarial attack framework, the practice is to set the node budget equal to its degree for local attacks [32], or choose a loss function which orders gradients in order of nodes closer to the decision boundary for global attacks [31]. However, due to the budget minimizing objective of DVM, we need to compute the budget precisely and pick candidate nodes which need the least budget.

Therefore, to compute the minimum budget $B^t(v)$ that converts node v, we use the bisection method [11, 12]. Algorithm 1 shows the entire pipeline. For each node, we compute the sorted gradients, $\hat{P}$ in Eq. 16 once and run bisection search over these gradients to find the minimal set of perturbations required to convert v. We initialize the lower and upper bound for search as 0 and $\deg(v)$, the degree of v, respectively. Thereafter, the upper bound is doubled until it is sufficient to convert v. We set the maximum upper bound equal to the maximum node degree of the network. After fixing the upper bound, bisection search repeatedly halves the search interval by checking feasibility of conversion at the midpoint of the interval and converges logarithmically.

We observe that other nodes can also flip due to the same structure or attribute changes made for flipping a candidate node. Therefore, for the purpose of choosing the best candidate node, we update the budget,

$$B^t(v) \leftarrow B^t(v) + (|S^{t-1}| - |S^t_v|) \tag{18}$$

where S_v^t is the set of adopters at time t if node v is selected as the candidate node to flip. Thus, $B^t(v)$ represents both the node flipping budget and the "collateral damage" to other nodes from flipping it. Therefore, if a node requires needs more budget but causes high collateral damage it is preferable to a node with a lesser actual budget.

Since budgets need to be recomputed for all non-adopters at every step, we make the algorithm faster by hashing node budgets and only recomputing budgets for nodes whose budget has changed. The recompute set R^t is defined as the set of nodes whose logit scores changed in the previous time step,

$$R^t = \{v | v \in D^t, z_v^t \neq z_v^{t-1}\} \tag{19}$$

For nodes whose logit scores are unchanged the actual budget might still change slightly, but for the purposes of picking the best candidate node we ignore these small changes.

G More details on Meta Influence and Meta Attribute Flips

Due to the dynamic sequence of changes involved in spreading product adoption in DVM, first-order gradients in Eq. 15 are insufficient to capture the long-range effects of a perturbation. While the flipping budget is minimized at each step, we need to characterize nodes that have high influence so that adoptions can sequentially cascade. Therefore, we develop the Meta Influence heuristic to model long-range effects and estimate downstream influence. Meta Influence uses Meta Attribute Flips which are feature perturbations that increase the potency of outgoing edge perturbations at an adopter node. Consequently, the Meta Influence is defined as the normalized gradient score on an adopter's outgoing edge perturbations post Meta Attribute Flips.

For Meta Attribute Flips, we again restrict feature perturbations, this time only to the features of node v, and define corresponding feature perturbation weights P_X,

$$x_v^{t'} = x_v^t + P_X \circ (\mathbf{1} - x_v^t) \tag{20}$$

where t' indicates an auxilliary time step and we initialize $P_X = \mathbf{0}$. To capture the effect of Meta Attribute Flips, we consider the following influence loss which is the sum of a CW-type loss [13] on all non-adopters nodes, and compute gradient scores $\hat{P}_X$,

$$L_{infl}^{t'}(v) = \sum_{w \in D} (z_{w,1}^t - z_{w,0}^t), \qquad \hat{P}_X^{t'} = \max\left(\left[\frac{\partial L_{infl}^{t'}(v)}{\partial P_X^{t'}}\right], 0\right) \tag{21}$$

Thereafter, we update the features x_v using Meta Attribute Flips, which are the top-k ranked perturbations in $P_X^{t'}$, for the purposes of computing Meta Influence,

$$i_X^{t'} = \text{argtop-k}\, \hat{P}_X^{t'}, \qquad [x_v^{t'}]_{i_X^{t'}} \leftarrow 1 \tag{22}$$

where k is a hyper-parameter controlling the number of Meta Attribute Flips. From Thm. 1, Meta Attribute Flips find feature changes that align with the GNN's classifier weights and increase the dynamic outgoing edge influence to other nodes in Eq. 7. Further, they also help to increase the margin from the GNN decision boundary and thus increase the node's potency.

For finding the Meta Influence, we restrict the outgoing edge perturbations P_A from node v to the non-adopters D, and define corresponding edge perturbation weights P_A,

$$A_{v,D}^{t''} = A_{v,D}^{t'} + P_A^{t''} \circ (\mathbf{1} - A_{v,D}^{t'}) \tag{23}$$

where t'' indicates another auxilliary time step after t' and we initialize $P_A^{t''} = \mathbf{0}$. We then consider the same influence loss in Eq. 21 on all non-adopters *at time* t'', and compute non-negative gradient scores $\hat{P}_A^{t''}$,

$$\hat{P}_A^{t''} = \max\left(\left[\frac{\partial L_{infl}^{t''}(v)}{\partial P_A^{t''}}\right], 0\right) \tag{24}$$

We note that herein the influence loss uses the perturbed feature $x_v^{t'}$, therefore Meta Influence takes into account second order gradient effects. However, the gradients $\hat{P}_A$ themselves are not second-order since $x_v^{t'}$ itself doesn't contain any first-order gradients. Finally, we denote the Meta Influence $I^t(v)$ of a node v as the normalized gradient score in $\hat{P}_A^{t''}$ averaged over the number of non-adopters,

$$I^t(v) = \frac{1^T \hat{P}_A^{t''}}{|D|} \tag{25}$$

During candidate node selection, Meta Influence is used for tiebreaking between equal budget nodes (Eq. 3). Further, we threshold on Meta Influence to perform Meta Attribute Flips at candidate nodes after flipping,

$$x_v^t \leftarrow \begin{cases} x_v^{t'} & \text{if } I^t(v) \geq \beta \\ x_v^t & \text{otherwise} \end{cases} \tag{26}$$

where β is a hyper-parameter controlling the threshold. Using Meta Influence, we can estimate which nodes will have high influence on their outgoing edges after Meta Attribute Flips, and thus judiciously allocate budget for Meta Attribute Flips.

H Complexity

In each step, for each target, DGI makes 1 backward pass to compute gradients and then sorts the gradients in $O(E \log E)$ time, followed by $O(\log \Delta)$ forward passes in bisection search, where Δ is the maximum allowed budget, which we set as the maximum degree of the graph. For the set of minimum budget nodes, the Meta Influence makes 2 forward and 2 backward passes. Time taken for a forward or backward pass in a 2-layer GNN on a GPU is $\mathcal{O}(1)$ assuming small input, hidden and output sizes. Thus, in the first step of DGI, we make $O(|D|)$ inner steps of gradient-guided node flipping, budget computation, meta attribute flips and meta influence, which takes a total of $O(|D|(E \log E + \log \Delta))$ time. In later steps, due to budget hashing, we only recompute budgets for $O(1)$ nodes, which takes $O(E \log E + \log \Delta)$. Assuming that every step of DGI flips $O(1)$ non-adopter nodes, the total runtime complexity of DGI to achieve a spread of ϕ can be determined,

$$O(\phi(E \log E + \log \Delta)) + O(|D|(E \log E + \log \Delta)) \tag{27}$$

Algorithm 1: Bisection search to find minimum budget.

Input: $G = (A, X)$, GNN f_θ, $\hat{P}$, Target node v.
Output: $B(v)$, minimum budget to flip v.

Init bounds $U = \deg(v)$ and $L = 0$
do
  $U = 2U$
while *v is not flipped by top-U perturbations in $\hat{P}$*;
do
  $C = \lfloor \frac{U+L}{2} \rfloor$
  if *v is not flipped by top-C perturbations in $\hat{P}$* **then**
    $L = C$
  else
    $U = C$
  end
while $U - L > 1$;
return : U

Algorithm 2: DGI

Input: G=(A,X), GNN f_θ, Adopters S, Non-Adopters D, k, β
Output: Minimum budget required s.t. $|S| = 500$

Init budget $\mathcal{B} = 0$
do
  for $v \in D$ **do**
    Compute $\hat{P}$ in Eq. 16
    Call Algorithm 1 to compute $B(v)$
  end
  Find N, the set of minimum budget nodes
  for $v \in N$ **do**
    Compute the Meta Influence $I(v)$
  end
  $v^* \leftarrow \max_{v \in N} I(v)$
  Perturb the network to flip v^*
  if $I(v^*) > \beta$ **then**
    Make k Meta Attribute Flips at v^*
    $B(v^*) \leftarrow B(v^*) + k$
  end
  $\mathcal{B} \leftarrow \mathcal{B} + B(v^*)$
  $S, D \leftarrow \{v|y'_v = 1\}, \{v|y'_v = 0\}$
while $|S| < 500$;
return : $\mathcal{B}$

I Experimental Details

Datasets. We utilize real-world attributed datasets to evaluate DGI and baseline approaches. Epinions and Ciao [14] are datasets collected from two popular product review sites, where each user can specify their trust relation in addition to rating products. Flixster [15] is a dataset collected from a popular movie rating website with an associated social graph. We create new splits for these datasets using the provided user networks and product ratings from [14, 28]. We sort the nodes using their degrees and take the subgraph corresponding to the lowest degree nodes. To generate features for each user, we pick a binary value for each product/movie based on whether they rated it or not. We choose the product/movie with the least number of seeds as the optimization goal for DVM. The dataset statistics are depicted in Table 2.

Table 2: Dataset statistics. $|\mathcal{S}|$ denotes seed set size.

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	Avg, Max Deg.	$\lvert\mathcal{S}\rvert$	Feats
Flixster	1045	1488	2.8, 153	5	839
Epinions	1054	1214	2.3, 25	5	2999
Ciao	1057	1190	2.3, 36	7	2999

Baselines: We compare to the following approaches:

- **Degree** selects target nodes based on the low-degree first heuristic. Until the target is flipped, Degree repeatedly spends a unit budget by first randomly picking a seeder node, then either adds a link from the seeder to the target if they aren't conencted, else turns on a feature with high-correlation to the label.
- **Margin** selects target nodes based on the low-margin first heuristic, i.e, nodes that have a smaller margin to the decision boundary are picked first. Edits to the structure and features are made the same as Degree.
- **GradArgmax** [25] is a gradient based white-box adversarial attack on structure. Target nodes are selected based on the low-loss heuristic, i.e, nodes with lower losses are picked first. We adapt GradArgmax to make edits to both structure and features.
- **MiBTack** [28] is another white-box adversarial attack which dynamically adjusts node budgets for topology-based PGD [27]. We adapt MiBTack to make edits to both structure and features while selecting target nodes the same as GradArgmax.

I.1 Implementation details

For the propagation model, we use 2-layer GNN architectures, as stacking multiple layers can lead to oversmoothing in GNNs [33]. We report results with both GCN [5] and GraphSAGE [21] as the backbone propagation models. We set the hidden layer to size 64 and use ReLU as the intermediate non-linear function. We train models using cross entropy loss for 200 epochs, a patience of 50, learning rate 1e-2 with cosine annealing and weight decay regularization 5e-4. We use all nodes and edges during training to attain the best GNN decision boundary. All models achieve 100% accuracy on the seed set along with a small number of false positives that are in the vicinity of the seeds and are included in the initial seed set. For the spread approaches, the hyperparameter k for number of Meta Attribute Flips is set to a value in $[1, 2, 4, 8, 16]$ using grid search with Fixed DGI. The threshold hyerparameter β is set to a value within $[0, 1]$ of the maximum Meta Influence in Fixed DGI using grid search with Dynamic DGI. The maximum upper bound to convert a node is set to the maximum degree in the graph during bisection search. If the spread approach is unable to increase the size of the adopters for 40 steps, we halt and report failure. We use spread approaches with GCN backbones for all our analysis, and note that the SAGE backbone yields the same insights. DGI and baselines are implemented using the DeepRobust library in Pytorch [34] for adversarial attacks on GNNs. All GNN models and spread approaches are trained and executed on a single NVIDIA RTX2080 GPU with 8GM RAM.

I.2 Cascades created by DGI

To understand cascades created by the DGI spread, in Figure. 4a, we qualitatively visualize a subgraph spanned by the Dynamic DGI edges on Flixster and color each node according to its cascade hop

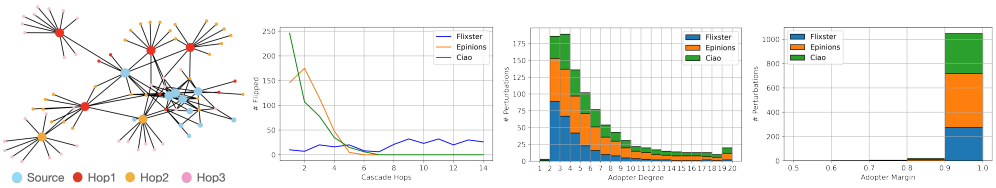


Figure 4: (a) Visualization of cascading dynamics of DGI. Node sizes and colors correspond to number of perturbations and cascade hops respectively. Only edges added by DGI are depicted. (b) Number of flipped nodes at different cascade hops. DGI creates long and staggered cascades for viral marketing. (c) Histogram of perturbations contributed by intermediary spreader nodes with increasing degree. (d) Histogram of perturbations contributed by intermediary spreader nodes with increasing GCN classification margin. Higher contributions are made by spreader nodes with low degrees and high margins.

distance from the initial seed set in this subgraph. We define the cascade hop distance of a node from the seed set inductively:

$$hop(i) = 1 + \max_{j \in PN(i)} hop(j) \tag{28}$$

where $hop(i) = 0$ for nodes in the initial seed set, and $PN(i)$ denotes perturbed neighbors of node i at the step when it flips. Further, we use node size to indicate the number of perturbations the node contributes in the course of the multi-step spread. We clearly see a strong pattern of cascading flips, whereby a node flipped earlier later flips many more and so on inductively. Therefore, the DGI spread creates cascading flipping, similar to a chain of referrals in a social network, where each referral is an entirely new edge. We also see from the node sizes that a few nodes are dominant spreaders while others contribute very little.

To understand the cascades created by DGI quantitatively, in Figure. 4b we plot the number of non-adopter flips with increasing hop distances for Flixster, Epinions and Ciao. Multi-hop cascade flips account for a sizable number of the total flips, which indicates that multi-hop path flips help in decreasing the budget required for the spread. Further, the cascade hop length can be considerably large. Particularly, for Flixster, we see nodes with cascade hop lengths up to 30, indicating how the added perturbations can percolate the adoption far from the seed set.

I.3 Characteristics of intermediary spreaders

To understand the characteristics of intermediary spreader nodes, we plot the histogram of perturbations contributed by intermediary spreader nodes with respect to their degree and classification margin in Figure. 4c and Figure. 4d respectively. For each dataset, we count the number of perturbations arising from nodes with the degree and classification margin lying within the same sub-interval. The degree and margin are considered at the moment the perturbation is made to account for dynamic changes. Due to the degree normalization in GNN message passing, low degree nodes have a higher influence edge weight, and we observe that nodes with low degree are highly correlated to higher number of perturbations. On the other hand, high classification margin indicates high feature and neighborhood alignment with the GNN classifier, therefore making outgoing edge or feature perturbations more potent. Thus, we see that perturbations are made exclusively at nodes with the maximum possible margin.