RETHINKING LEARNING-BASED INFLUENCE MAXIMIZATION: SIMPLE NEURAL SURROGATES AND NATIVE DISCRETE SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing learning-based influence maximization frameworks rely heavily on complex neural architectures and continuous optimization over seed representations. We challenge this paradigm with SIMBA, a diffusion-model-agnostic framework pairing a lightweight neural surrogate with direct discrete search. SIMBA consists of three key components: 1) uniformly anchored node embeddings that eliminate initialization noise and encourage learning driven by graph topology and diffusion patterns, 2) a shallow two-layer graph neural network surrogate predicting final infection states, and 3) batched multi-swap simulated annealing that explores combinatorial seed space without gradients or continuous relaxation. By shifting compute from complex representation learning to effective discrete search, SIMBA drastically cuts time-to-solution while achieving superior influence spread and data efficiency.

1 INTRODUCTION

Influence Maximization (IM), the task of identifying a small subset of seed nodes that maximizes the expected spread of information over a network, is a fundamental problem in network science, viral marketing, and epidemiology (Kempe et al., 2003; Chen et al., 2010; Neophytou et al., 2024). A growing body of work has explored learning-based methods that leverage reinforcement learning, graph neural networks, and other learned representations to improve influence estimation and seed selection (Panagopoulos et al., 2020; Li et al., 2022; Chen et al., 2023; Zhang et al., 2026a). Frameworks such as DeepIM (Ling et al., 2023) and DeepSN (Hevapathige et al., 2025) demonstrate that influence spread can be predicted without explicit knowledge of the underlying diffusion model, using only observed pairs of seed configurations and their corresponding diffusion outcomes to train neural surrogates. These neural surrogates can then replace expensive diffusion simulations, enabling diffusion-model-agnostic IM on large graphs with state-of-the-art performance. Despite this progress, existing neural surrogate frameworks share an implicit assumption: effective diffusion-model-agnostic IM requires complex neural architectures together with continuous optimization over seed representations.

In this work, we challenge this assumption and propose SIMBA (**S**urrogate-guided **I**nfluence **M**aximization with **B**atched simulated **A**nnealing), which consists of three complementary components. First, we introduce *uniformly anchored node embeddings*, where every learnable node embedding is initialized from a shared random anchor, and the input seed configuration is encoded by masking non-seed embeddings. This design removes initialization noise and allows message passing to construct node representations purely from graph topology and diffusion patterns. Second, we employ a *lightweight neural surrogate* implemented using a two-layer graph neural network (GNN) that predicts the final infection state of every node given an initial seed configuration. Finally, we formulate IM as a discrete optimization problem and solve it using a *batched multi-swap simulated annealing* algorithm that repeatedly queries the learned surrogate. Since candidate seed sets are evaluated in parallel using batched inference, the optimization process efficiently explores the discrete search space without requiring repeated diffusion simulations or gradient computations.

An important consequence of this framework is a shift in computational effort from expensive representation learning to effective combinatorial search. Existing approaches invest substantial computation in learning expressive continuous representations to enable efficient gradient-based inference. In con-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

trast, the lightweight surrogate employed by SIMBA converges rapidly, allowing significantly more computation to be devoted to high-quality discrete search. As a result, the overall time-to-solution is substantially reduced, while the output seed sets consistently yield significantly higher influence.

Our experiments further reveal an unexpected degree of data efficiency. We find that a small fraction of seed and diffusion outcome pairs used by existing neural surrogate frameworks is sufficient for SIMBA to learn an effective surrogate. The resulting reduction in training cost, together with more aggressive surrogate-guided search, allows the framework to scale efficiently to graphs containing hundreds of thousands of nodes while achieving state-of-the-art performance.

Finally, we investigate why such a lightweight surrogate generalizes effectively to previously unseen seed configurations. Through an analysis of the learned embedding space, we observe that uniformly anchored node embeddings often naturally evolve into a highly structured low-dimensional geometric manifold after learning. This emergent organization produces smooth surrogate predictions for novel combinations of seed nodes, providing a geometric explanation for the strong generalization performance observed during discrete optimization.

Our contributions are summarized as follows.

- We propose SIMBA, a diffusion-model-agnostic IM framework that combines a lightweight GNN surrogate with batched multi-swap simulated annealing to search over the discrete seed space.
- We introduce uniformly anchored node embeddings that encode seed sets through embedding masking and encourage representation learning driven by graph topology and diffusion patterns.
- We demonstrate that combining a lightweight surrogate with direct discrete optimization achieves higher influence spread and significantly lower time-to-solution than existing IM frameworks.
- We show that SIMBA remains highly data efficient, requiring only a small fraction of diffusion observations while maintaining strong performance on large-scale graphs.
- We provide an empirical analysis of the learned node embedding geometry, offering insight into why the proposed surrogate generalizes effectively to previously unseen seed configurations.

2 Related Work

Traditional IM methods typically offer principled optimization and provable approximation bounds but often require full access to the underlying diffusion model and are limited to specific diffusion models (Kempe et al., 2003; Leskovec et al., 2007; Saito et al., 2012; Borgs et al., 2014; Tang et al., 2014; 2015; Nguyen et al., 2016; Li et al., 2019b; Hong et al., 2020). In empirical real-world networks, the underlying propagation rules are rarely known or easily parameterized, severely restricting the applicability of these approaches.

There is a growing literature proposing learning-based IM approaches. Most such approaches rely on reinforcement learning to train an agent that selects the next seed node (Lin et al., 2015; Li et al., 2019a; Tian et al., 2020; Manchanda et al., 2020; Li et al., 2022; Chen et al., 2023). Since the reward is the marginal influence gain, these methods need to repeatedly observe the diffusion outcomes. There are also methods that directly train GNNs to predict the influence spread. Panagopoulos et al. (2020) leverage fine-grained diffusion cascades as the training data; Kumar et al. (2022) expect every node's influence as the training label. Addressing the performance limitations and scalability issues of all aforementioned methods, DeepIM (Ling et al., 2023) and DeepSN (Hevapathige et al., 2025) utilize continuous optimization and sophisticated GNN architectures to train surrogate models that predict the influence spread of given seed sets and search for high-quality seed sets through gradient-based optimization. We provide a detailed review of DeepIM and DeepSN in Appendix A. Most recently, Zhang et al. (2026b) attempt to achieve cross-graph generalization; Zhang et al. (2026a) incorporate more information into their model by observing the node states at each diffusion step. However, both of these works require access to the underlying diffusion models to run simulations for model training.

3 Problem formulation

Let $G = (V, E)$ denote a directed graph with node set $V = \{v_1, v_2, \ldots, v_N\}$ and edge set E. The objective of IM is to identify a seed set $S \subset V$ with a fixed size $|S| = k$ that maximizes the expected information spread over the network. We denote the influence spread of a seed set by $\sigma(S)$, which represents the expected number of nodes activated under the underlying diffusion process. We

consider a *diffusion-model-agnostic* setting, where the objective is to learn the diffusion dynamics directly from observed diffusion outcomes without access to the underlying propagation mechanism.

Specifically, given a graph G, we assume access to a training dataset $\mathcal{D} = \{(S_m, \mathbf{y}_m)\}_{m=1}^{M}$, where each seed set S_m is represented by a binary indicator vector $\mathbf{s}_m \in \{0,1\}^N$. The corresponding target $\mathbf{y}_m \in \{0,1\}^N$ denotes the observed final infection state after diffusion, where $y_{m,i} = 1$ indicates that node v_i is activated and $y_{m,i} = 0$ otherwise. The goal of diffusion-model-agnostic IM is therefore to learn a surrogate mapping $f_\theta(G, \mathbf{s}) = \hat{\mathbf{y}}$, where $\hat{\mathbf{y}} \in \mathbb{R}^N$ approximates the final infection state produced by the unknown diffusion process. The predicted influence of a candidate seed set can then be estimated as $\hat{\sigma}(S) = \sum_{i=1}^{N} \hat{y}_i$. Using this learned surrogate, the original IM problem becomes $S^* = \arg\max_{|S|=k} \hat{\sigma}(S)$, which can be solved without explicitly simulating the unknown diffusion process.

4 METHOD

Figure 1 presents an overview of SIMBA. We detail its key components in the following subsections.

4.1 UNIFORMLY ANCHORED NODE EMBEDDINGS

Given a graph $G = (V, E)$, we associate every node v_i with a learnable embedding $\mathbf{e}_i \in \mathbb{R}^d$. This improves the expressiveness of the model but introduces the cold start problem on unseen nodes (i.e., $v_i \notin \cup_{m=1}^{M} S_m$). To resolve this, we introduce a simple uniform initialization anchor. Specifically, all node embeddings are initialized from the same random anchor: $\mathbf{e}_i = \mathbf{c}, \forall v_i \in V$, where $\mathbf{c} \in \mathbb{R}^d$ is sampled once at initialization. By enforcing an identical initial coordinate across all node embeddings, we isolate the GNN from feature noise. Freed from initialization noise, the node embedding updates are driven by the graph topology and diffusion patterns. The anchor also acts as a special token that is easier for the model to interpret compared to unseen random embeddings.

For a candidate seed set represented by a binary vector $\mathbf{s} \in \{0,1\}^N$, we construct the node feature matrix by masking the embeddings according to the seed configuration: $\mathbf{x}_i = s_i \mathbf{e}_i$ (i.e., $\mathbf{x}_i = \mathbf{e}_i$ if $v_i \in S$, and $\mathbf{0}$ otherwise). This formulation allows the same learnable node representations to be reused across arbitrary seed configurations. The seed set is introduced only through masking, while the GNN learns how diffusion propagates from activated nodes through graph topology and diffusion outcomes.

4.2 LIGHTWEIGHT NEURAL SURROGATE

The goal of the neural surrogate is to approximate the unknown diffusion process by predicting the final infection state of every node from an initial seed configuration via a surrogate mapping $f_\theta(G, \mathbf{s}) = \hat{\mathbf{y}}$. We instantiate f_θ using a lightweight two-layer GraphSAGE network (Hamilton et al., 2017). The input node features are the uniformly anchored node embeddings introduced in the previous section after masking according to the input seed configuration. Through neighborhood aggregation, the GNN propagates information from the activated seed nodes across the graph, allowing the network to approximate the underlying diffusion dynamics without requiring explicit knowledge of the diffusion model. We intentionally adopt a shallow and unconstrained architecture, as our objective is to learn a fast yet sufficiently accurate surrogate for efficient seed set search.

The surrogate is trained using the observation pairs in $\mathcal{D}$ by minimizing the mean squared error between the predicted and observed final infection states: $\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$. After training, the surrogate parameters are frozen. For a candidate seed set S, the predicted influence is computed by aggregating the predicted node states, $\hat{\sigma}(S) = \sum_{i=1}^{N} \hat{y}_i$, which serves as the objective function for the downstream optimization algorithm. Consequently, seed set optimization requires only repeated forward passes through the lightweight surrogate, eliminating the need for gradient-based optimization, latent seed representations, or repeated diffusion simulations during inference.

4.3 BATCHED MULTI-SWAP SIMULATED ANNEALING

After training the surrogate, IM is formulated as the discrete optimization problem $S^* = \arg\max_{|S|=k} \hat{\sigma}(S)$. To solve this problem efficiently, we introduce a batched multi-swap simulated annealing procedure. Instead of optimizing a single seed set trajectory, we simultaneously maintain a batch of B candidate solutions: $\mathcal{S} = \{S_1, S_2, \ldots, S_B\}$. Each candidate satisfies

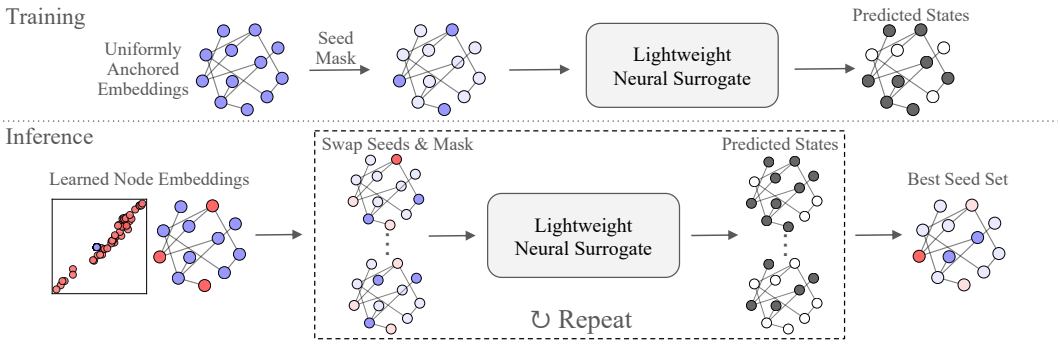


Figure 1: Overview of SIMBA. Red: seen nodes; blue: uniform anchor; low opacity: masked nodes.

the cardinality constraint $|S_b| = k$. At each iteration, a neighboring solution is generated for every candidate by replacing r currently selected nodes with r unselected nodes. Specifically, for candidate S_b, we sample $R_b \subset S_b, A_b \subset V \setminus S_b$, with $|R_b| = |A_b| = r$, and construct $S'_b = (S_b \setminus R_b) \cup A_b$. Because the numbers of removals and additions are identical, every candidate remains a valid size-k seed set. All neighboring solutions are evaluated simultaneously through a single batched forward pass: $\hat{\sigma}(S'_1), \ldots, \hat{\sigma}(S'_B) = f_\theta(\mathcal{S}')$. For each trajectory, we compute $\Delta_b = \hat{\sigma}(S'_b) - \hat{\sigma}(S_b)$. The proposed solution is accepted according to the simulated annealing rule: $p_{\text{acc}} = 1$ if $\Delta_b > 0$, and $\exp(\Delta_b/T)$ otherwise, where T is the current temperature. The temperature is updated after each iteration using geometric cooling: $T_{i+1} = \alpha T_i$, where α is the cooling rate. Throughout optimization, we maintain the best solution observed across all parallel search trajectories. Algorithm 1 summarizes the proposed optimization procedure.

Algorithm 1 Batched Multi-Swap Simulated Annealing Guided by Neural Surrogate

Require: Trained surrogate f_θ, graph G, seed budget k, number of steps I, batch size B, number of swaps r, initial temperature T_0, cooling rate α

Initialize B random seed sets $\{S_1, \ldots, S_B\}$ with $|S_b| = k$
Evaluate $\hat{\sigma}(S_b)$ for all b using batched surrogate inference
Initialize S^* as the highest-scoring candidate
$T \leftarrow \alpha T_0$
for $i = 1, \ldots, I$ **do**
  for each candidate S_b **do**
    Sample r nodes $R_b \subset S_b$ to remove
    Sample r nodes $A_b \subset V \setminus S_b$ to add
    Construct neighbor: $S'_b = (S_b \setminus R_b) \cup A_b$
  end for
  Evaluate all neighbors using f_θ with batched inference
  for each candidate S_b **do**
    Compute $\Delta_b = \hat{\sigma}(S'_b) - \hat{\sigma}(S_b)$
    if $\Delta_b > 0$ or $\text{rand}() < \exp(\Delta_b/T)$ **then**
      Accept S'_b as the new candidate
    end if
  end for
  Update S^* with the best candidate found so far
  $T \leftarrow \alpha T$
end for
return S^*

4.4 Computational efficiency

The computational philosophy of SIMBA differs fundamentally from existing learning-based IM frameworks. Previous approaches invest the majority of their computational budget in offline surrogate learning, employing complex neural architectures and continuous latent optimization to minimize the cost of seed set optimization during inference. In contrast, SIMBA intentionally redistributes computation by learning a lightweight neural surrogate and allocating the majority of the computational budget to surrogate-guided discrete search during inference. This design substantially reduces training overhead while enabling more extensive exploration of the combinatorial seed space.

From a computational perspective, both the surrogate learning stage and the discrete optimization stage scale linearly with the size of the graph. To further improve inference efficiency, SIMBA employs two complementary strategies. First, candidate seed sets are evaluated in parallel using batched inference, allowing multiple search trajectories to share a single forward pass through the surrogate. Second, the proposed multi-swap neighborhood operator explores larger regions of the discrete search space at each iteration, reducing the number of optimization steps required to reach high-quality solutions compared with conventional single-swap local search.

Additionally, we uncover and exploit an unexpected data efficiency property of diffusion-model-agnostic IM. As validated in Section 5.3, the surrogate can be trained using only a small fraction of the

available diffusion observations while maintaining comparable IM performance. This substantially reduces surrogate training time and enables more computational resources to be devoted to discrete optimization. On very large graphs, where both training and optimization become increasingly expensive, this data efficiency and more exploratory search lead to significant reductions in time-to-solution while preserving the quality of the discovered seed sets.

5 EXPERIMENTS

Datasets. We evaluate SIMBA on five real-world benchmark graphs, Jazz, Network Science (NS), Power Grid (PG) (Rossi & Ahmed, 2015), Cora-ML (McCallum et al., 2000), and Digg (Lerman et al., 2012), and one synthetic Erdős–Rényi graph (Ling et al., 2023) spanning a wide range of network sizes and structural characteristics. We consider the linear threshold (LT) (Granovetter, 1978), independent cascade (IC) (Kempe et al., 2003), and susceptible-infected-susceptible (Kermack & McKendrick, 1927) (SIS) diffusion models. Dataset details are provided in Appendix B.

Baselines. We compare against *traditional IM* methods, including IMM (Tang et al., 2015), OPIM (Tang et al., 2018), and SubSIM (Guo et al., 2020), an *online IM* method, OIM (Lei et al., 2015), and *learning-based IM* methods, including IMINFECTOR (Panagopoulos et al., 2020), PIANO (Li et al., 2022), ToupleGDD (Chen et al., 2023), DeepIM (Ling et al., 2023) and DeepSN (Hevapathige et al., 2025). Additionally, we include DynaFLUX (Zhang et al., 2026a) in Table 8 in Appendix E.3.

Evaluation metrics. Following prior works, the primary evaluation metric is the achieved influence spread under the underlying diffusion process with 1%, 5%, 10%, and 20% of the nodes as seed nodes. We provide a note on how the reported spread is computed in Appendix C. We additionally report the time-to-solution (TTS), defined as the combined surrogate training time and surrogate-guided search time required to produce the final seed set.

Implementation details. Unless otherwise specified, the node embedding dimension is fixed at $d = 8$. The neural surrogate consists of two GraphSAGE layers (Hamilton et al., 2017) followed by a linear prediction head. Both GraphSAGE layers use a hidden dimension of 128, and a residual connection (He et al., 2016) is added to the second layer. All models are trained for 200 epochs using AdamW (Kingma & Ba, 2015; Loshchilov & Hutter, 2019) with a learning rate of $5e^{-4}$ and a weight decay of $1e^{-4}$. The same model architecture and training hyperparameters are used throughout all experiments. Additional implementation details are provided in Appendix D.

5.1 INFLUENCE MAXIMIZATION PERFORMANCE

We first evaluate whether the proposed framework is capable of identifying high-quality seed sets despite employing a lightweight neural surrogate and direct discrete optimization. We consider three SIMBA variants: 1) the constrained setting (C) where the search is only done over nodes seen during surrogate training, 2) the unconstrained setting (U) where the search space equals the entire node set, and 3) the two-stage constrained-to-unconstrained setting (U*) where the search is first done over the seen nodes, and the resulting seed sets are used as the initializations for the following unconstrained search. Table 1 compares the influence spread obtained by SIMBA across all benchmark datasets. As noted by Zhang et al. (2026a), the evaluation protocol of DeepSN[1] for the SIS experiments is different from that of other baselines: it runs 200 steps of simulation as opposed to 100. Thus, we also exclude DeepSN from comparison for the SIS experiments.

Overall, SIMBA consistently achieves the strongest performance, obtaining the highest influence spread in 62 of the 72 (86.1%) settings. The improvements are frequently substantial rather than marginal, indicating that a lightweight neural surrogate coupled with direct discrete search can outperform considerably more sophisticated learning-based frameworks. These results suggest that the primary limitation of existing learning-based approaches is not the expressive power of the surrogate model, but rather the optimization strategy used to identify seed sets, and that neither highly expressive GNN architectures nor continuous optimization are prerequisites for effective IM. Although SIMBA does not achieve the best result in every setting, the remaining cases exhibit relatively small performance differences, indicating that the proposed framework remains highly competitive across diverse graph structures and diffusion scenarios.

[1] https://github.com/Aselahp/DeepSN.

Table 1: Influence spread (%) under the IC, LT, and SIS models. The best (resp. second best) results are in bold (resp. underlined). Δ is the difference between SIMBA and the best baseline.

	Cora-ML (IC)				Network Science (IC)				Power Grid (IC)				Jazz (IC)				Synthetic (IC)				Digg (IC)			
Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
IMM	8.1	26.2	37.3	50.2	5.2	16.8	27.0	45.7	4.3	17.4	31.5	51.1	2.6	20.1	31.4	42.8	9.2	26.2	36.3	51.6	7.4	18.4	32.8	49.6
OPIM	13.4	26.9	37.4	50.9	6.6	19.4	28.9	48.6	5.7	17.7	29.7	50.1	2.4	20.1	34.4	46.8	9.6	25.3	36.6	51.7	7.6	18.5	32.9	48.9
SubSIM	10.1	25.7	36.8	51.1	4.8	15.4	27.9	44.8	4.6	19.2	31.7	50.2	3.6	18.8	37.6	44.7	9.5	26.7	36.5	51.5	7.5	18.9	33.3	49.4
OIM	8.9	27.6	38.0	51.3	4.2	16.7	26.5	48.2	5.7	17.5	31.9	50.8	2.0	18.5	36.3	42.2	9.6	26.2	36.7	51.3	7.8	18.2	33.1	49.6
IMINFECTOR	9.6	26.8	37.7	50.6	5.4	17.9	27.8	47.6	5.4	18.2	31.6	50.9	3.6	19.7	37.5	45.9	9.1	26.2	36.1	51.5	7.9	18.6	33.5	49.8
PIANO	9.8	25.2	37.4	51.1	4.7	16.3	27.1	47.2	5.3	18.1	31.7	50.2	2.2	19.2	36.6	43.2	9.1	26.4	36.2	51.6	–	–	–	–
ToupleGDD	10.6	27.5	38.5	51.5	6.3	17.8	28.3	50.5	5.4	19.3	31.6	51.3	3.3	20.4	37.2	45.7	9.5	26.8	37.1	51.4	–	–	–	–
DeepIM	14.1	28.1	39.6	52.4	7.8	20.9	31.5	51.2	6.3	21.0	32.5	52.4	4.9	23.3	41.5	49.9	11.6	27.4	38.7	52.1	8.4	19.3	34.2	51.3
DeepSN	11.5	25.6	40.9	52.8	6.2	22.0	32.0	52.4	6.4	24.0	36.9	**61.0**	8.5	26.9	**41.6**	**53.8**	10.6	27.8	38.8	52.8	8.9	19.5	35.2	52.8
SIMBA (C)	14.2	33.2	39.8	54.7	9.5	24.0	36.6	50.3	**8.5**	26.7	40.5	57.6	10.4	27.2	36.7	50.0	11.9	28.3	38.5	52.2	60.3	66.3	72.3	80.6
SIMBA (U)	16.9	33.3	43.1	57.3	**9.7**	24.0	36.6	57.8	8.2	26.2	40.4	59.5	**12.3**	**27.6**	37.5	51.1	11.6	**28.4**	**39.4**	**53.8**	**60.4**	**66.5**	72.3	80.6
SIMBA (U*)	**17.2**	**33.6**	**43.3**	**57.4**	9.6	**24.4**	**37.1**	**58.1**	**8.5**	**26.9**	**41.3**	60.1	11.7	26.7	37.5	50.8	**12.0**	28.3	**39.4**	**53.8**	60.3	66.4	**72.5**	**81.2**
Δ	+3.1	+5.5	+2.4	+4.6	+1.9	+2.4	+5.1	+5.7	+2.1	+2.9	+4.4	-0.9	+3.8	+0.7	-4.1	-2.7	+0.4	+0.6	+0.6	+1.0	+51.5	+47.0	+37.3	+28.4

	Cora-ML (LT)				Network Science (LT)				Power Grid (LT)				Jazz (LT)				Synthetic (LT)				Digg (LT)			
Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
IMM	1.7	34.8	52.2	66.4	2.5	11.9	18.1	33.6	4.6	19.9	31.7	56.9	1.4	5.7	13.4	24.5	1.1	5.2	13.1	66.9	2.4	10.8	37.4	55.6
OPIM	2.3	36.9	51.2	71.5	1.6	12.0	18.8	34.1	4.4	21.6	29.4	55.5	1.4	6.9	12.6	20.9	1.3	5.2	12.6	62.1	2.1	11.3	38.2	57.1
SubSIM	1.7	33.6	54.7	70.1	1.8	10.4	19.2	34.1	4.5	21.1	31.2	57.4	1.4	5.9	11.4	21.2	1.4	5.5	13.1	69.6	2.4	11.3	37.9	56.9
IMINFECTOR	2.1	33.9	51.3	70.6	2.1	11.8	18.7	34.5	4.2	21.3	31.6	56.2	1.4	6.2	13.5	22.8	1.3	5.2	12.9	67.4	2.2	11.1	38.9	58.7
PIANO	2.1	33.5	53.3	69.8	2.1	11.3	19.1	33.9	4.3	21.3	31.4	57.1	1.1	6.2	12.1	22.4	1.2	5.2	12.9	67.4	–	–	–	–
ToupleGDD	2.3	36.2	54.5	70.9	2.8	12.4	19.8	34.6	4.8	21.9	32.6	58.1	1.4	6.5	12.9	23.6	1.3	5.5	13.4	70.2	–	–	–	–
DeepIM	13.4	69.2	83.5	94.1	4.1	16.6	26.7	41.5	6.3	24.4	46.8	71.7	1.9	6.5	16.4	99.1	1.5	6.5	15.5	99.9	3.5	15.9	41.3	76.2
DeepSN	7.4	40.7	68.2	95.3	2.9	14.4	25.3	52.0	6.3	24.6	47.2	73.2	**2.0**	6.7	14.9	96.9	**1.6**	5.8	13.5	99.9	3.2	16.1	41.7	72.1
SIMBA (C)	**20.7**	76.7	84.8	89.4	4.9	22.1	28.6	41.9	**7.8**	**32.5**	54.2	77.4	1.3	7.4	**71.2**	98.6	1.1	6.5	15.9	**100.0**	12.3	46.4	85.0	90.6
SIMBA (U)	18.9	80.0	**93.3**	98.8	6.8	**26.5**	**41.7**	66.4	7.6	31.0	58.4	88.1	**2.0**	**30.4**	48.0	**100.0**	1.4	7.0	15.9	**100.0**	15.4	81.9	85.2	90.9
SIMBA (U*)	18.0	**80.3**	93.0	**98.9**	**7.0**	26.3	41.6	**66.6**	**7.8**	32.4	**59.5**	**88.7**	**2.0**	22.5	58.5	**100.0**	1.4	**7.1**	**16.4**	**100.0**	**18.6**	**82.4**	**86.3**	**91.8**
Δ	+7.3	+11.1	+9.8	+3.6	+2.9	+9.9	+15.0	+14.6	+1.5	+7.9	+12.3	+15.5	0.0	+23.5	+54.8	+0.9	-0.2	+0.6	+0.9	+0.1	+15.1	+66.3	+44.6	+15.6

	Cora-ML (SIS)				Network Science (SIS)				Power Grid (SIS)				Jazz (SIS)				Synthetic (SIS)				Digg (SIS)			
Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
IMM	2.0	9.5	15.4	27.6	1.3	5.6	12.2	22.1	1.1	5.6	11.0	22.9	7.6	37.8	55.6	67.1	2.7	12.6	20.9	37.3	2.5	9.4	16.3	32.6
OPIM	2.3	9.3	16.2	27.2	1.4	5.9	13.0	22.1	1.2	5.9	11.2	22.4	5.7	44.7	58.6	68.3	2.8	12.5	20.2	36.1	2.3	9.3	16.5	32.1
SubSIM	2.3	9.2	16.9	28.8	1.5	5.6	12.2	23.3	1.2	5.6	11.4	21.9	2.9	30.1	53.8	67.0	2.5	12.6	20.2	36.5	2.5	9.5	16.1	32.3
IMINFECTOR	2.1	9.4	16.1	27.9	1.7	5.8	12.4	22.3	1.3	5.5	12.4	23.1	8.8	35.4	54.8	66.2	2.5	12.4	20.5	36.2	2.3	9.1	16.4	32.4
DeepIM	7.1	**16.1**	21.9	30.8	**2.7**	8.7	15.1	25.1	**1.9**	7.6	13.3	23.8	27.1	**57.1**	**68.1**	**74.1**	3.2	14.4	24.5	39.1	5.6	11.4	18.8	**36.3**
SIMBA (C)	**7.2**	15.7	21.9	31.1	2.5	8.8	14.6	23.8	**1.9**	**7.8**	**14.1**	24.7	**33.6**	55.4	63.3	70.3	**3.5**	**14.6**	24.8	39.7	**16.5**	20.3	**24.9**	33.5
SIMBA (U)	**7.2**	15.7	**22.5**	**32.5**	2.4	8.8	**15.3**	**26.5**	**1.9**	**7.8**	14.0	25.1	30.8	56.7	64.7	71.6	3.4	14.3	24.7	40.2	**16.5**	20.3	**24.9**	33.5
SIMBA (U*)	**7.2**	15.8	**22.5**	32.4	2.4	**8.9**	**15.3**	**26.5**	**1.9**	**7.8**	**14.1**	**25.2**	32.1	56.6	64.3	71.2	**3.5**	**14.6**	**24.9**	**40.4**	**16.5**	**20.4**	**24.9**	33.6
Δ	+0.1	-0.3	+0.6	+1.7	-0.2	+0.2	+0.2	+1.4	0.0	+0.2	+0.8	+1.4	+6.5	-0.4	-3.4	-2.5	+0.3	+0.2	+0.4	+1.3	+10.9	+9.0	+6.1	-2.7

Moreover, expanding the search space consistently improves IM performance. The unconstrained strategy (U) is no worse than the constrained setting (C) in 58 of the 72 (80.6%) settings, meaning that the learned surrogate generalizes effectively to previously unseen nodes. If the surrogate failed to produce reliable predictions outside the seen nodes, enlarging the search space would introduce noisy objective evaluations and degrade optimization performance. Instead, the consistent improvements indicate that the surrogate successfully extrapolates beyond the node configurations encountered during training. The two-stage strategy (U*) further strengthens this claim. By first identifying a high-quality solution within the constrained search space and subsequently refining it over the full node set, U* is no worse than C in 66 of the 72 (91.7%) settings. Since every refinement step is guided solely by the surrogate's predictions, the consistent improvement establishes that the surrogate provides sufficiently reliable evaluations of previously unseen seed configurations to guide effective optimization. These results provide strong empirical evidence that SIMBA achieves robust generalization, allowing the surrogate to meaningfully evaluate seed sets containing nodes that were never explored during training.

5.2 TIME-TO-SOLUTION

High influence quality alone is insufficient for practical deployment. Existing learning-based IM frameworks often achieve fast inference only after substantial offline computation devoted to training. Consequently, reporting inference time alone provides an incomplete picture of the overall computational cost. As DeepIM runs much faster than earlier learning-based frameworks (Ling et al., 2023), our runtime analysis focuses on DeepIM[2], DeepSN, and SIMBA.

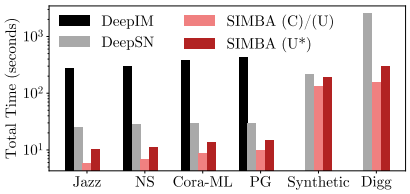


Figure 2: TTS across datasets.

Table 2 reports the TTS breakdown, and Figure 2 compares the TTS of all approaches. Despite allocating more computation to inference for discrete optimization, SIMBA achieves significantly

[2]https://github.com/triplej0079/DeepIM.

Table 2: TTS breakdown (training/inference).

	Jazz	Network Science	Cora-ML	Power Grid	Synthetic	Digg
DeepIM	274.33s (272.64s/1.69s)	300.58s (298.80s/1.78s)	376.69s (374.80s/1.89s)	426.59s (424.70s/1.89s)	–	–
DeepSN	24.62s (23.54s/1.09s)	28.44s (24.49s/3.95s)	29.50s (28.02s/1.49s)	29.01s (27.28s/1.73s)	219.19s (211.40s/7.79s)	2552.14s (2243.58s/308.56s)
SIMBA (C)/(U)	5.81s (0.72s/5.09s)	6.89s (1.67s/5.22s)	8.62s (2.91s/5.71s)	9.78s (4.06s/5.72s)	130.80s (69.77s/61.03s)	*154.86s* (*8.91s*/*145.95s*)
SIMBA (U*)	10.09s (0.72s/9.37s)	11.32s (1.67s/9.65s)	13.52s (2.91s/10.62s)	14.71s (4.06s/10.65s)	190.96s (69.77s/121.19s)	*299.85s* (*8.91s*/*290.94s*)

Table 3: Influence spread (%) achieved by SIMBA with varying fractions of training data.

		Cora-ML (IC)				Network Science (IC)				Power Grid (IC)				Jazz (IC)			
Training data	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
1%	SIMBA (U)	16.8	32.2	42.8	57.2	9.7	23.8	37.0	58.0	7.9	25.8	40.2	59.3	12.1	27.5	37.5	49.3
5%	SIMBA (U)	16.9	32.5	43.2	56.8	9.7	24.0	36.8	58.0	8.2	25.8	40.4	59.5	12.3	27.5	37.1	50.2
20%	SIMBA (U)	16.9	32.4	43.1	57.0	9.7	23.9	37.1	58.0	7.9	25.9	40.6	59.4	11.7	26.6	36.3	50.3
100%	SIMBA (U)	16.9	33.3	43.1	57.3	9.7	24.0	36.6	57.8	8.2	26.2	40.4	59.5	12.3	27.6	37.5	51.1
		Cora-ML (LT)				Network Science (LT)				Power Grid (LT)				Jazz (LT)			
Training data	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
1%	SIMBA (U)	18.1	80.8	93.4	98.8	6.9	26.1	42.4	66.8	7.6	30.9	58.0	88.4	2.0	23.0	58.7	100.0
5%	SIMBA (U)	19.0	80.7	92.8	98.9	6.9	25.9	41.8	66.2	7.6	30.6	58.3	88.3	2.0	22.7	56.8	100.0
20%	SIMBA (U)	19.3	80.7	93.0	98.7	6.8	26.3	42.3	67.0	7.6	30.7	58.2	88.1	2.0	30.4	54.5	100.0
100%	SIMBA (U)	18.9	80.0	93.3	98.8	6.8	26.5	41.7	66.4	7.6	31.0	58.4	88.1	2.0	30.4	48.0	100.0
		Cora-ML (SIS)				Network Science (SIS)				Power Grid (SIS)				Jazz (SIS)			
Training data	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
1%	SIMBA (U)	7.0	15.5	21.9	31.6	2.4	9.0	14.6	25.4	1.8	7.4	13.4	24.3	35.2	56.1	64.6	71.7
5%	SIMBA (U)	7.0	15.4	22.5	32.3	2.3	9.1	15.1	25.9	1.8	7.5	13.5	24.7	32.8	56.6	64.7	71.4
20%	SIMBA (U)	7.1	15.8	22.4	32.4	2.4	8.8	15.2	26.0	1.8	7.6	13.8	25.0	28.9	56.6	64.9	71.2
100%	SIMBA (U)	7.2	15.7	22.5	32.5	2.4	8.8	15.3	26.5	1.9	7.8	14.0	25.1	30.8	56.7	64.7	71.6

lower TTS because its lightweight surrogate trains rapidly and each optimization iteration requires only a batched forward pass through the fixed network. However, as shown in Table 2, as the graph size increases (from Jazz to Synthetic), the training time of SIMBA grows much faster than the inference time and gradually takes over. Additionally, because the search space grows exponentially with respect to the graph size, a large number of search steps is necessary to find high-quality seed sets on large graphs. For these two reasons, the TTS advantage of SIMBA over DeepSN is diminishing.

To improve the TTS scalability of SIMBA, we implement two strategies for experiments on Digg that target the training and inference stages, respectively. First, we reduce the number of training samples from 80 to 2. Second, we reduce the number of search steps from 50,000 to 20,000 and increase the number of swaps from 1 to 20 to enable more exploratory search. Table 9 in Appendix E.4 compares the influence spread achieved by the two search parameter settings. The resulting TTS of SIMBA on Digg is up to an order of magnitude faster than DeepSN and can remain comparable to that on Synthetic, as shown in Table 2, highlighting the exceptional TTS scalability of SIMBA, which is further illustrated in Figure 3.

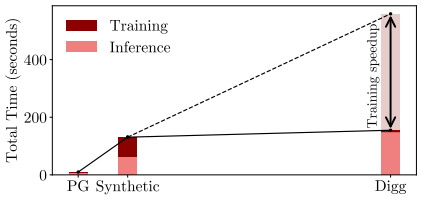


Figure 3: TTS scalability of SIMBA.

5.3 Data efficiency

Obtaining diffusion observations is another dominant practical bottleneck in learning-based IM. We therefore investigate how the performance of SIMBA varies as the amount of available training data decreases. Table 3 reports the influence spread obtained when training the surrogate using different fractions of the available observation pairs. The full set of results is provided in Appendix E.5. Surprisingly, we observe that performance saturates rapidly, with only a small fraction of the training observations required to recover nearly the same influence spread as models trained on the complete dataset.

This result has two important implications. First, the proposed surrogate exhibits strong sample efficiency, substantially reducing the cost of collecting diffusion observations. Second, the resulting

Table 4: Influence spread (%) achieved by SIMBA with different node embedding initialization strategies and post-training embedding update rules.

		Cora-ML (IC)				Network Science (IC)				Power Grid (IC)				Jazz (IC)			
Setting	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
Random, no update	SIMBA (U)	16.4	32.4	42.4	56.3	9.4	23.7	36.5	56.9	7.9	25.7	39.8	58.7	10.4	25.3	36.1	50.5
Random, weighted avg	SIMBA (U)	14.8	32.4	38.9	53.0	9.2	23.7	36.0	49.6	7.9	25.4	38.5	55.0	10.6	26.8	36.3	49.4
Random, most similar	SIMBA (U)	16.7	32.5	42.7	56.2	9.3	**24.0**	**36.6**	56.7	7.8	25.5	39.8	58.8	9.7	26.8	36.6	50.5
Random, global mean	SIMBA (U)	14.4	32.3	38.6	53.3	9.3	23.7	36.1	49.4	8.0	25.3	38.8	54.7	10.4	26.5	35.9	49.4
Anchored, no update	SIMBA (U)	**16.9**	**33.3**	**43.1**	**57.3**	**9.7**	**24.0**	**36.6**	**57.8**	**8.2**	**26.2**	**40.4**	**59.5**	**12.3**	**27.6**	**37.5**	**51.1**
		Cora-ML (LT)				Network Science (LT)				Power Grid (LT)				Jazz (LT)			
Setting	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
Random, no update	SIMBA (U)	15.3	79.1	90.9	97.6	5.6	22.0	35.2	59.4	6.6	26.7	52.7	83.1	1.8	7.7	24.0	99.5
Random, weighted avg	SIMBA (U)	16.9	74.1	84.0	89.3	4.7	21.0	29.8	43.0	7.0	29.2	51.3	75.8	1.9	7.1	**68.6**	94.9
Random, most similar	SIMBA (U)	15.3	78.0	90.9	97.4	6.6	22.0	37.6	61.7	7.0	27.3	54.0	85.0	**2.0**	13.1	24.5	98.8
Random, global mean	SIMBA (U)	16.1	73.7	84.4	89.6	4.9	21.1	28.9	42.3	7.1	29.8	51.1	75.6	1.9	7.1	66.4	99.5
Anchored, no update	SIMBA (U)	**18.9**	**80.0**	**93.3**	**98.8**	**6.8**	**26.5**	**41.7**	**66.4**	**7.6**	**31.0**	**58.4**	**88.1**	**2.0**	**30.4**	48.0	**100.0**
		Cora-ML (SIS)				Network Science (SIS)				Power Grid (SIS)				Jazz (SIS)			
Setting	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
Random, no update	SIMBA (U)	6.8	15.5	22.5	**32.9**	2.1	8.1	14.5	25.8	1.6	7.2	13.3	24.4	31.6	54.6	62.8	70.8
Random, weighted avg	SIMBA (U)	**7.2**	15.8	21.9	31.2	**2.9**	8.9	14.4	23.6	1.8	7.7	13.8	24.2	32.3	54.0	61.6	69.9
Random, most similar	SIMBA (U)	7.1	15.7	**22.6**	32.6	2.7	**9.0**	15.3	26.2	1.8	7.4	13.6	24.4	31.1	53.2	62.7	70.0
Random, global mean	SIMBA (U)	**7.2**	**15.8**	21.8	31.2	**2.9**	8.8	14.5	23.6	1.8	7.7	13.8	24.2	**32.7**	54.2	61.5	68.5
Anchored, no update	SIMBA (U)	**7.2**	15.7	22.5	32.5	2.4	8.8	**15.3**	**26.5**	**1.9**	**7.8**	**14.0**	**25.1**	30.8	**56.7**	**64.7**	**71.6**

reduction in surrogate training time enables a larger fraction of the computational budget to be allocated toward discrete search. On very large graphs, this redistribution of computation leads to significant reductions in total runtime (see Figure 3) while maintaining competitive IM performance.

5.4 Mechanistic analysis of uniformly anchored node embeddings

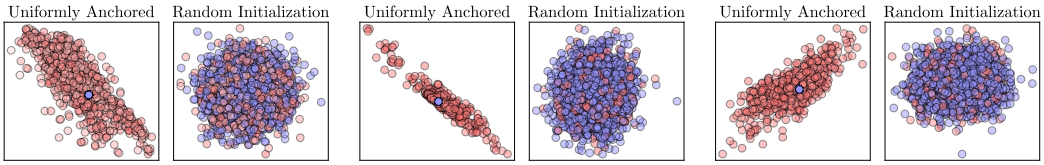
The previous experiments establish that SIMBA performs well empirically. We now investigate why such a lightweight surrogate generalizes effectively to previously unseen seed configurations. We also showcase the architectural robustness of SIMBA by evaluating an extremely lightweight variant where each node is represented using only its binary seed indicator in Appendix E.9.

Effect of initialization strategy on seed set quality. The proposed framework initializes every node embedding from a common random anchor before learning topology and diffusion-dependent representations. To evaluate the importance of this design, we compare against conventional independent random initialization, where each node begins from a distinct embedding. We additionally consider three post-training update strategies for handling unseen nodes: 1) computing a graph structural feature similarity weighted average over the embeddings of seen nodes, 2) assigning the embedding of the single most structurally similar seen node, and 3) assigning the global mean of the embeddings of seen nodes. For the graph structural features, we include degree centrality, PageRank (Page et al., 1998), clustering coefficient (Watts & Strogatz, 1998), betweenness centrality (Freeman, 1977), and core number (Batagelj & Zaversnik, 2003). Weights and similarities are computed using cosine similarity.

Table 4 shows that uniformly anchored initialization consistently produces higher downstream influence spread. The full set of results is provided in Appendix E.6. While unseen node update strategies may improve the performance of independently initialized embeddings, the uniformly anchored approach achieves the strongest overall performance without requiring any post hoc operation. Furthermore, as revealed in Table 12 in Appendix E.7, when employing uniformly anchored node embeddings, there is no clear benefit of utilizing custom update rules for the unseen nodes; this design already robustly handles them.

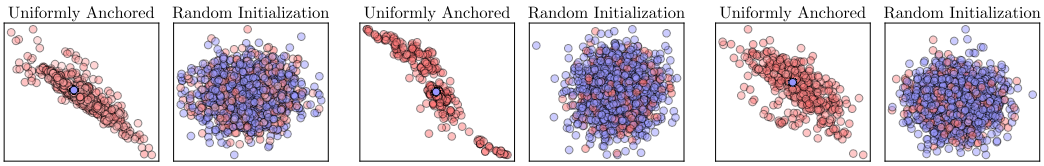
Geometric interpretation of the embedding space. To further understand the learned representations, we visualize the embedding space produced by surrogate training under different initialization strategies. For direct interpretability, we set the node embedding dimension to $d = 2$, allowing us to plot the learned latent space directly without relying on dimensionality reduction. We confirm in Table 13 in Appendix E.8 that the $d = 2$ variant maintains high downstream IM performance.

As shown in Figures 4, 5, 6, and 7, uniformly anchored node embeddings often naturally evolve into a smooth, highly structured low-dimensional manifold after training, forming an elliptical distribution,



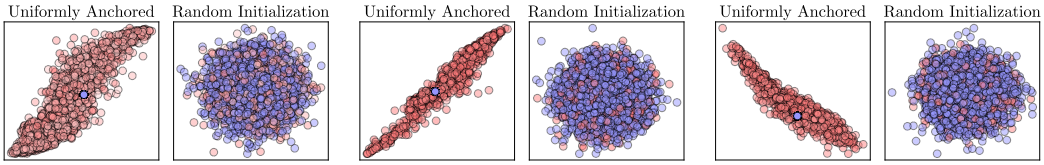
(a) IC (b) LT (c) SIS

Figure 4: Learned node embeddings of Cora-ML. Red: seen nodes; blue: unseen nodes.



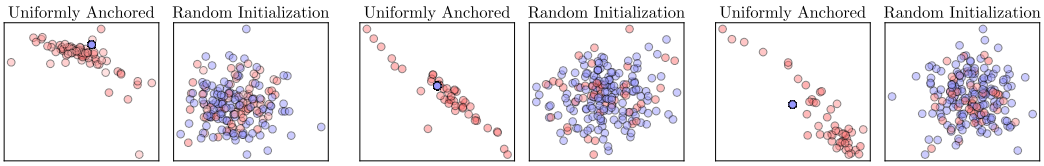
(a) IC (b) LT (c) SIS

Figure 5: Learned node embeddings of Network Science. Red: seen nodes; blue: unseen nodes.



(a) IC (b) LT (c) SIS

Figure 6: Learned node embeddings of Power Grid. Red: seen nodes; blue: unseen nodes.



(a) IC (b) LT (c) SIS

Figure 7: Learned node embeddings of Jazz. Red: seen nodes; blue: unseen nodes.

with the embeddings of the unseen nodes centered. Because every node starts from the same initialization anchor, learning is driven purely by graph topology and diffusion patterns. In contrast, independently initialized embeddings form an isotropic distribution whose higher variance results in a less structured embedding distribution, with the embeddings of the seen and unseen nodes scattered.

We identify two advantages of the resulting embedding space geometry: 1) The low-dimensional manifold supports linear scalability. When the GNN aggregates neighborhoods, the embeddings naturally scale linearly, giving a strong and explicit signal of local influence density; 2) Acting as a special token, the anchor forces the embeddings of the unseen nodes to have the same values, which encourages the active neighbors of such nodes to dictate their role in influence spread prediction and makes it easier for the GNN to meaningfully override their embeddings through message passing. Together, these geometric properties induced by the uniform anchor prevent downstream discrete search from encountering uncalibrated local maxima, explaining why a lightweight GNN surrogate can support robust combinatorial optimization in discrete space.

6 Conclusion

In this work, we presented `SIMBA`, a paradigm for IM that shifts computational effort from training complex model architectures to direct discrete search. By pairing uniformly anchored node embeddings with a shallow two-layer GNN surrogate, `SIMBA` robustly predicts complex infection dynamics in a single forward pass without requiring explicit parameterization of propagation rules. Combined with surrogate-guided batched multi-swap simulated annealing, the framework directly explores the combinatorial seed space, completely bypassing gradient computations and continuous relaxations. Empirical results demonstrate that `SIMBA` drastically cuts overall time-to-solution while consistently achieving superior influence spread and remarkable data efficiency.

AI USE STATEMENT

In this work, we used generative AI tools to create and edit software code and to edit the manuscript to improve readability. We reviewed all AI-assisted work. All AI-assisted code was verified and tested for correctness, and all AI-assisted manuscript edits were reviewed by the authors for accuracy and consistency with the intended content. We take responsibility for the final content of this work, including the text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Vladimir Batagelj and Matjaz Zaversnik. An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*, 2003.

Cristian Bodnar, Francesco Di Giovanni, Benjamin Chamberlain, Pietro Lio, and Michael Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. *Advances in Neural Information Processing Systems*, 35:18527–18541, 2022.

Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. Maximizing social influence in nearly optimal time. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pp. 946–957. SIAM, 2014.

Tiantian Chen, Siwen Yan, Jianxiong Guo, and Weili Wu. Touplegdd: A fine-designed solution of influence maximization by deep reinforcement learning. *IEEE Transactions on Computational Social Systems*, 11(2):2210–2221, 2023.

Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1029–1038, 2010.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Matthias Fey, Jinu Sunil, Akihiro Nitta, Rishi Puri, Manan Shah, Blaž Stojanovič, Ramona Bendias, Alexandria Barghi, Vid Kocijan, Zecheng Zhang, Xinwei He, Jan Eric Lenssen, and Jure Leskovec. Pyg 2.0: Scalable learning on real world graphs. In *Temporal Graph Learning Workshop @ KDD 2025*, 2025.

Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pp. 35–41, 1977.

Mark Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, 83(6): 1420–1443, 1978. ISSN 00029602, 15375390.

Qintian Guo, Sibo Wang, Zhewei Wei, and Ming Chen. Influence maximization revisited: Efficient reverse reachable set generation with bound tightened. In *Proceedings of the 2020 ACM SIGMOD international conference on management of data*, pp. 2167–2181, 2020.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Jakob Hansen and Thomas Gebhart. Sheaf neural networks. In *TDA & Beyond*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Asela Hevapathige, Qing Wang, and Ahad N Zehmakan. Deepsn: a sheaf neural framework for influence maximization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(16): 17177–17185, 2025.

Wenjing Hong, Chao Qian, and Ke Tang. Efficient minimum cost seed selection with theoretical guarantees for competitive influence maximization. *IEEE transactions on cybernetics*, 51(12): 6091–6104, 2020.

David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 137–146, 2003.

W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 115(772):700–721, 1927. ISSN 09501207.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Sanjay Kumar, Abhishek Mallik, Anavi Khetarpal, and Bhawani Sankar Panda. Influence maximization in social networks using graph embedding and graph neural network. *Information Sciences*, 607:1617–1636, 2022.

Siyu Lei, Silviu Maniu, Luyi Mo, Reynold Cheng, and Pierre Senellart. Online influence maximization. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 645–654, 2015.

Kristina Lerman, Rumi Ghosh, and Tawan Surachawala. Social contagion: An empirical study of information spread on digg and twitter follower graphs. *arXiv preprint arXiv:1202.3162*, 2012.

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 420–429, 2007.

Hui Li, Mengting Xu, Sourav S Bhowmick, Changsheng Sun, Zhongyuan Jiang, and Jiangtao Cui. Disco: Influence maximization meets network embedding and deep learning. *arXiv preprint arXiv:1906.07378*, 2019a.

Hui Li, Mengting Xu, Sourav S Bhowmick, Joty Shafiq Rayhan, Changsheng Sun, and Jiangtao Cui. Piano: Influence maximization meets deep reinforcement learning. *IEEE Transactions on Computational Social Systems*, 10(3):1288–1300, 2022.

Xiang Li, J David Smith, Thang N Dinh, and My T Thai. Tiptop:(almost) exact solutions for influence maximization in billion-scale networks. *IEEE/ACM Transactions on Networking*, 27(2):649–661, 2019b.

Su-Chen Lin, Shou-De Lin, and Ming-Syan Chen. A learning-based framework to handle multi-round multi-party influence maximization on social networks. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 695–704, 2015.

Chen Ling, Junji Jiang, Junxiang Wang, My T Thai, Renhao Xue, James Song, Meikang Qiu, and Liang Zhao. Deep graph representation learning and optimization for influence maximization. In *International conference on machine learning*, pp. 21350–21361. PMLR, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Sahil Manchanda, Akash Mittal, Anuj Dhawan, Sourav Medya, Sayan Ranu, and Ambuj Singh. Gcomb: Learning budget-constrained combinatorial algorithms over billion-sized graphs. *Advances in Neural Information Processing Systems*, 33:20000–20011, 2020.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

Nicola Neophytou, Afaf Taïk, and Golnoosh Farnadi. Promoting fair vaccination strategies through influence maximization: a case study on covid-19 spread. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(20):22285–22293, 2024.

Hung T Nguyen, My T Thai, and Thang N Dinh. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In *Proceedings of the 2016 international conference on management of data*, pp. 695–710, 2016.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bring order to the web. In *Proc. of the 7th International World Wide Web Conf*, 1998.

George Panagopoulos, Fragkiskos D Malliaros, and Michalis Vazirgiannis. Multi-task learning for influence estimation and maximization. *IEEE Transactions on Knowledge and Data Engineering*, 34(9):4398–4409, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: an imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019.

Ryan Rossi and Nesreen Ahmed. The network data repository with interactive graph analytics and visualization. *Proceedings of the AAAI conference on artificial intelligence*, 29(1), 2015.

Kazumi Saito, Masahiro Kimura, Kouzou Ohara, and Hiroshi Motoda. Efficient discovery of influential nodes for sis models in social networks. *Knowledge and information systems*, 30(3): 613–635, 2012.

Jing Tang, Xueyan Tang, Xiaokui Xiao, and Junsong Yuan. Online processing algorithms for influence maximization. In *Proceedings of the 2018 international conference on management of data*, pp. 991–1005, 2018.

Youze Tang, Xiaokui Xiao, and Yanchen Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 75–86, 2014.

Youze Tang, Yanchen Shi, and Xiaokui Xiao. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pp. 1539–1554, 2015.

Shan Tian, Songsong Mo, Liwei Wang, and Zhiyong Peng. Deep reinforcement learning-based approach to tackle topic-aware influence maximization. *Data Science and Engineering*, 5(1):1–11, 2020.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393 (6684):440–442, 1998.

Daiyunke Zhang, Ting Deng, Tianchen Zhu, Shuai Ma, Daqing Li, Mingtian Peng, and Feng Tian. Dynaflux: Implicit dynamics-preserving reinforcement learning for topology-free influence maximization. In *Proceedings of the ACM Web Conference 2026*, pp. 4940–4951, 2026a.

Haotian Zhang, Kai Han, Zhizhuo Yin, Shuang Cui, Jing Tang, and Pan Hui. Imgnn: An efficient, effective and generalizable algorithm for influence maximization in social networks. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 1904–1915, 2026b.

A ADDITIONAL LITERATURE REVIEW

DeepIM. DeepIM (Ling et al., 2023) formulates IM as a continuous optimization problem. It employs an autoencoder to map discrete seed vectors into a continuous latent representation. In parallel, it trains a graph attention network (Veličković et al., 2018) to predict diffusion outcomes from binary seed configurations. During inference, it optimizes seed probabilities in the latent space through gradient updates to maximize the predicted final spread, projects the latent representation back to a seed vector, and applies a top-k projection to recover a discrete seed set. Although this approach enables efficient inference, the continuous latent seed set representations introduce additional learning overhead and a gap between continuous latent variables and discrete seed selections.

DeepSN. DeepSN (Hevapathige et al., 2025) follows a similar surrogate-based optimization framework but focuses on improving diffusion prediction accuracy through a more sophisticated graph architecture. Specifically, it replaces conventional message-passing networks with a sheaf neural network (Hansen & Gebhart, 2020; Bodnar et al., 2022) that takes in binary seed set vectors with one-hot-coded unique node degrees and incorporates an additional learnable node scoring mechanism during inference. While these architectural improvements enhance predictive capacity, they also introduce training complexity.

Limitations of existing approaches. Although existing neural surrogate frameworks demonstrate the feasibility of diffusion-model-agnostic IM, they rely on two implicit assumptions. First, they assume that high-quality seed set optimization requires continuous relaxation of the discrete search space. Second, they assume that increasingly complex neural architectures are necessary to accurately model diffusion dynamics.

In contrast, `SIMBA` investigates whether a lightweight neural surrogate combined with direct discrete optimization can achieve competitive or superior performance. Rather than optimizing a continuous representation of seed sets, `SIMBA` operates directly in the original combinatorial space using batched multi-swap simulated annealing guided by the learned surrogate. Our design removes latent seed representations and specialized architectures while enabling efficient and effective exploration of large-scale seed spaces.

B DATASETS

Table 5: Dataset statistics. The number of message-passing edges is reported.

	Jazz	NS	Cora-ML	PG	Synthetic	Digg
Nodes	198	1,589	2,810	4941	50,000	279,630
Edges	5,484	5,484	15,962	13188	500,000	3,096,252

For diffusion observation generation and IM performance evaluation, we run simulations for 100 diffusion steps on the graphs with each of the diffusion models and record the final node states as the diffusion outcomes. For the LT model, the thresholds are sampled uniformly from $[0.3, 0.6]$; for the IC model, the propagation probabilities are set to be $p_{u,v} = 1/d_v^{\text{in}}, \forall (u, v) \in E$, where d_v^{in} is the in-degree of node v; for the SIS model, the infection and recovery probabilities are both set to be 0.001. The diffusion configurations and simulation settings are consistent with prior works (Ling et al., 2023; Hevapathige et al., 2025; Zhang et al., 2026a).

C A NOTE ON FINAL SPREAD COMPUTATION FOR EVALUATION

The final spread is obtained by running simulations 100 times with the seed sets produced by the methods and averaging the results, which we view as one round of simulations. However, there can still be noticeable variance across rounds based on our observation. To further reduce the effect of randomness, we perform 5 rounds of such simulations (i.e., 5×100) and report the average over the 5 rounds (i.e., the average of the average spread). The results showing standard deviations are provided in Appendix E.1.

D ADDITIONAL IMPLEMENTATION DETAILS

The search hyperparameters are selected according to the graph size and search complexity. For Jazz, Cora-ML, Network Science, and Power Grid, we set the number of steps to $I = 10,000$, the batch size to $B = 4$, and the number of swaps $r = 1$. For Synthetic, we increase the search budget to $I = 50{,}000$ and decrease the batch size to $B = 1$ while keeping $r = 1$. For Digg, we use $I = 20{,}000$, $B = 1$, and $r = 20$ to enable more aggressive exploration of the substantially larger search space.

Given the number of sources of randomness and the probabilistic nature of the components in the pipeline (data split, model and search initializations, and diffusion simulation), we set the initial temperature $T_0 = 1e^{-6}$ and the cooling rate $\alpha = 1$ for all experiments included in the main paper to avoid further conflating with the effect of random candidate solution acceptance. We report the results with a different set of temperature and cooling parameters in Appendix E.2.

All models are implemented using PyTorch (Paszke et al., 2019) and PyTorch Geometric (Fey & Lenssen, 2019; Fey et al., 2025) libraries. Experiments are conducted on a machine with a single NVIDIA GeForce RTX 4090 GPU, a 32-core Intel Core i9-14900K CPU, and 64 GB of RAM running Ubuntu 24.04.

E ADDITIONAL EXPERIMENTAL RESULTS

E.1 INFLUENCE MAXIMIZATION PERFORMANCE OF SIMBA WITH STANDARD DEVIATIONS

Table 6: Influence spread (%) achieved by SIMBA with standard deviations.

	Cora-ML (IC)				Network Science (IC)			
Method	1%	5%	10%	20%	1%	5%	10%	20%
SIMBA(C)	14.2 ± 0.2	33.2 ± 0.2	39.8 ± 0.1	54.7 ± 0.1	9.5 ± 0.1	24.0 ± 0.2	36.6 ± 0.3	50.3 ± 0.1
SIMBA (U)	16.9 ± 0.1	33.3 ± 0.3	43.1 ± 0.3	57.3 ± 0.2	9.7 ± 0.2	24.0 ± 0.2	36.6 ± 0.6	57.8 ± 0.2
SIMBA (U*)	17.2 ± 0.2	33.6 ± 0.2	43.3 ± 0.3	57.4 ± 0.0	9.6 ± 0.3	24.4 ± 0.2	37.1 ± 0.4	58.1 ± 0.2
	Power Grid (IC)				Jazz (IC)			
Method	1%	5%	10%	20%	1%	5%	10%	20%
SIMBA(C)	8.5 ± 0.1	26.7 ± 0.2	40.5 ± 0.2	57.6 ± 0.1	10.4 ± 1.6	27.2 ± 1.1	36.7 ± 0.6	50.0 ± 0.5
SIMBA (U)	8.2 ± 0.1	26.2 ± 0.1	40.4 ± 0.3	59.5 ± 0.2	12.3 ± 0.8	27.6 ± 1.1	37.5 ± 0.6	51.1 ± 0.7
SIMBA (U*)	8.5 ± 0.1	26.9 ± 0.2	41.3 ± 0.3	60.1 ± 0.1	11.7 ± 0.4	26.7 ± 0.9	37.5 ± 0.9	50.8 ± 0.6
	Cora-ML (LT)				Network Science (LT)			
Method	1%	5%	10%	20%	1%	5%	10%	20%
SIMBA(C)	20.7 ± 0.4	76.7 ± 0.2	84.8 ± 0.1	89.4 ± 0.1	4.9 ± 0.3	22.1 ± 0.0	28.6 ± 0.1	41.9 ± 0.0
SIMBA (U)	18.9 ± 0.9	80.0 ± 1.1	93.3 ± 0.4	98.8 ± 0.1	6.8 ± 0.2	26.5 ± 0.3	41.7 ± 1.0	66.4 ± 0.9
SIMBA (U*)	18.0 ± 1.2	80.3 ± 0.6	93.0 ± 0.5	98.9 ± 0.1	7.0 ± 0.3	26.3 ± 0.0	41.6 ± 1.1	66.6 ± 1.0
	Power Grid (LT)				Jazz (LT)			
Method	1%	5%	10%	20%	1%	5%	10%	20%
SIMBA(C)	7.8 ± 0.1	32.5 ± 0.3	54.2 ± 0.1	77.4 ± 0.1	1.3 ± 0.0	7.4 ± 0.1	71.2 ± 1.3	98.6 ± 0.3
SIMBA (U)	7.6 ± 0.1	31.0 ± 0.4	58.4 ± 0.5	88.1 ± 0.5	2.0 ± 0.0	30.4 ± 0.2	48.0 ± 15.3	100.0 ± 0.0
SIMBA (U*)	7.8 ± 0.1	32.4 ± 0.4	59.5 ± 0.5	88.7 ± 0.6	2.0 ± 0.0	22.5 ± 10.9	58.5 ± 0.2	100.0 ± 0.0
	Cora-ML (SIS)				Network Science (SIS)			
Method	1%	5%	10%	20%	1%	5%	10%	20%
SIMBA(C)	7.2 ± 0.1	15.7 ± 0.1	21.9 ± 0.1	31.1 ± 0.1	2.5 ± 0.1	8.8 ± 0.2	14.6 ± 0.0	23.8 ± 0.1
SIMBA (U)	7.2 ± 0.1	15.7 ± 0.1	22.5 ± 0.1	32.5 ± 0.1	2.4 ± 0.1	8.8 ± 0.2	15.3 ± 0.2	26.5 ± 0.2
SIMBA (U*)	7.2 ± 0.1	15.8 ± 0.1	22.5 ± 0.1	32.4 ± 0.1	2.4 ± 0.1	8.9 ± 0.2	15.3 ± 0.1	26.5 ± 0.2
	Power Grid (SIS)				Jazz (SIS)			
Method	1%	5%	10%	20%	1%	5%	10%	20%
SIMBA(C)	1.9 ± 0.0	7.8 ± 0.0	14.1 ± 0.0	24.7 ± 0.1	33.6 ± 1.4	55.4 ± 0.4	63.3 ± 0.4	70.3 ± 0.3
SIMBA (U)	1.9 ± 0.0	7.8 ± 0.0	14.0 ± 0.1	25.1 ± 0.1	30.8 ± 2.6	56.7 ± 0.3	64.7 ± 0.3	71.6 ± 0.2
SIMBA (U*)	1.9 ± 0.0	7.8 ± 0.0	14.1 ± 0.0	25.2 ± 0.1	32.1 ± 2.4	56.6 ± 0.5	64.3 ± 0.3	71.2 ± 0.5

E.2 INFLUENCE MAXIMIZATION PERFORMANCE OF SIMBA WITH DIFFERENT TEMPERATURE SCHEDULES

Table 7: Influence spread (%) achieved by SIMBA with different temperature schedules.

		Cora-ML (IC)				Network Science (IC)				Power Grid (IC)				Jazz (IC)			
Setting	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
$T_0 = 100$ $\alpha = 0.2$	SIMBA (C)	14.2	33.2	40.0	54.5	9.4	23.8	36.7	50.3	8.4	26.6	40.6	57.8	9.8	27.1	36.5	50.2
	SIMBA (U)	16.7	33.2	43.1	56.9	**9.7**	24.2	37.1	57.8	8.2	26.1	40.6	59.5	12.1	27.3	**37.7**	50.1
	SIMBA (U*)	16.7	**33.6**	43.2	57.1	9.6	24.3	**37.2**	**58.2**	8.4	26.8	**41.4**	**60.1**	11.8	27.3	37.1	50.7
$T_0 = 1e^{-6}$ $\alpha = 1$	SIMBA (C)	14.2	33.2	39.8	54.7	9.5	24.0	36.6	50.3	**8.5**	26.7	40.5	57.6	10.4	27.2	36.7	50.0
	SIMBA (U)	16.9	33.3	43.1	57.3	**9.7**	24.0	36.6	57.8	8.2	26.2	40.4	59.5	**12.3**	**27.6**	37.5	**51.1**
	SIMBA (U*)	**17.2**	**33.6**	**43.3**	**57.4**	9.6	**24.4**	37.1	58.1	**8.5**	**26.9**	41.3	**60.1**	11.7	26.7	37.5	50.8

		Cora-ML (LT)				Network Science (LT)				Power Grid (LT)				Jazz (LT)			
Setting	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
$T_0 = 100$ $\alpha = 0.2$	SIMBA (C)	**20.9**	76.7	84.8	89.4	4.8	21.9	28.6	41.9	**7.8**	32.4	54.2	77.4	1.3	7.4	70.3	98.0
	SIMBA (U)	18.6	80.5	**93.4**	98.7	6.8	26.0	41.7	66.5	7.7	30.7	57.8	88.3	**2.0**	**30.4**	58.1	**100.0**
	SIMBA (U*)	19.6	**80.6**	**93.4**	98.7	6.9	26.2	**41.8**	66.5	**7.8**	32.1	**59.5**	**88.8**	**2.0**	**30.4**	59.6	**100.0**
$T_0 = 1e^{-6}$ $\alpha = 1$	SIMBA (C)	20.7	76.7	84.8	89.4	4.9	22.1	28.6	41.9	**7.8**	**32.5**	54.2	77.4	1.3	7.4	**71.2**	98.6
	SIMBA (U)	18.9	80.0	93.3	98.8	6.8	**26.5**	41.7	66.4	7.6	31.0	58.4	88.1	**2.0**	**30.4**	48.0	**100.0**
	SIMBA (U*)	18.0	80.3	93.0	**98.9**	**7.0**	26.3	41.6	**66.6**	**7.8**	32.4	**59.5**	88.7	**2.0**	22.5	58.5	**100.0**

		Cora-ML (SIS)				Network Science (SIS)				Power Grid (SIS)				Jazz (SIS)			
Setting	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
$T_0 = 100$ $\alpha = 0.2$	SIMBA (C)	**7.2**	15.8	21.9	31.1	**2.6**	8.8	14.7	23.8	**1.9**	**7.8**	**14.1**	24.7	**34.0**	55.2	63.1	69.9
	SIMBA (U)	7.1	15.7	22.4	**32.7**	2.4	8.8	15.4	26.3	**1.9**	7.7	13.9	25.1	30.4	56.1	64.5	71.3
	SIMBA (U*)	**7.2**	**15.9**	22.4	32.6	2.4	8.8	**15.5**	26.4	**1.9**	**7.8**	**14.1**	**25.2**	29.9	56.4	64.3	71.3
$T_0 = 1e^{-6}$ $\alpha = 1$	SIMBA (C)	**7.2**	15.7	21.9	31.1	2.5	8.8	14.6	23.8	**1.9**	**7.8**	**14.1**	24.7	33.6	55.4	63.3	70.3
	SIMBA (U)	**7.2**	15.7	**22.5**	32.5	2.4	8.8	15.3	**26.5**	**1.9**	**7.8**	14.0	25.1	30.8	**56.7**	**64.7**	**71.6**
	SIMBA (U*)	**7.2**	15.8	**22.5**	32.4	2.4	**8.9**	15.3	**26.5**	**1.9**	**7.8**	**14.1**	**25.2**	32.1	56.6	64.3	71.2

E.3 COMPARISON WITH DYNAFLUX

Table 8: Influence spread (%) achieved by SIMBA and DynaFLUX.

	Cora-ML (IC)				Network Science (IC)				Power Grid (IC)				Jazz (IC)				Synthetic (IC)				Digg (IC)			
Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
DynaFLUX	35.3	46.8	55.5	65.4	7.8	21.1	29.6	41.7	9.0	27.4	41.3	58.6	26.0	37.3	41.7	56.4	**14.4**	**30.4**	41.1	54.4	60.3	66.2	74.8	**84.6**
DynaFLUX+	**38.4**	**47.4**	**56.7**	**68.2**	9.3	23.4	32.9	45.1	**9.1**	**29.0**	**43.1**	**61.6**	**26.3**	**39.7**	**45.2**	**60.8**	14.1	30.3	**41.4**	**54.5**	**60.8**	**68.6**	**74.1**	78.4
SIMBA (C)	14.2	33.2	39.8	54.7	9.5	24.0	36.6	50.3	8.5	26.7	40.5	57.6	10.4	27.2	36.7	50.0	11.9	28.3	38.5	52.2	60.3	66.3	72.3	80.6
SIMBA (U)	16.9	33.3	43.1	57.3	**9.7**	24.0	36.6	57.8	8.2	26.2	40.4	59.5	12.3	27.6	37.5	51.1	11.6	28.4	39.4	53.8	60.4	66.5	72.3	80.6
SIMBA (U*)	17.2	33.6	43.3	57.4	9.6	**24.4**	**37.1**	**58.1**	8.5	26.9	41.3	60.1	11.7	26.7	37.5	50.8	12.0	28.3	39.4	53.8	60.3	66.4	72.5	81.2

	Cora-ML (LT)				Network Science (LT)				Power Grid (LT)				Jazz (LT)				Synthetic (LT)				Digg (LT)			
Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
DynaFLUX	15.1	73.8	89.5	94.4	5.0	20.3	30.8	45.0	8.3	31.5	58.3	83.2	1.5	7.1	36.9	99.9	1.0	5.2	11.2	99.9	79.6	85.4	**91.9**	**97.1**
DynaFLUX+	15.3	73.7	90.0	96.2	6.0	23.7	34.8	49.5	**8.7**	**34.9**	**62.7**	88.4	1.6	7.6	14.2	99.9	1.1	5.3	11.4	99.9	**80.9**	**87.6**	90.3	91.7
SIMBA (C)	**20.7**	76.7	84.8	89.4	4.9	22.1	28.6	41.9	7.8	32.5	54.2	77.4	1.3	7.4	**71.2**	98.6	1.1	6.5	15.9	**100.0**	12.3	46.4	85.0	90.6
SIMBA (U)	18.9	80.0	**93.3**	98.8	6.8	**26.5**	**41.7**	66.4	7.6	31.0	58.4	88.1	**2.0**	**30.4**	48.0	**100.0**	**1.4**	7.0	15.9	**100.0**	15.4	81.9	85.2	90.9
SIMBA (U*)	18.0	**80.3**	93.0	**98.9**	**7.0**	26.3	41.6	**66.6**	7.8	32.4	59.5	**88.7**	**2.0**	22.5	58.5	**100.0**	**1.4**	**7.1**	**16.4**	**100.0**	18.6	82.4	86.3	91.8

	Cora-ML (SIS)				Network Science (SIS)				Power Grid (SIS)				Jazz (SIS)				Synthetic (SIS)				Digg (SIS)			
Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
DynaFLUX	6.9	15.2	20.8	29.9	2.3	7.0	12.4	22.7	1.8	6.8	12.2	22.3	30.6	56.4	64.5	70.3	3.0	12.6	22.2	37.0	16.5	18.0	21.1	28.5
DynaFLUX+	**7.3**	**15.9**	**22.5**	31.9	**2.8**	8.8	14.1	24.2	**2.3**	7.6	13.7	**25.4**	**35.8**	**58.9**	**64.7**	68.6	3.2	13.5	23.6	38.4	**17.1**	**20.4**	23.7	31.7
SIMBA (C)	7.2	15.7	21.9	31.1	2.5	8.8	14.6	23.8	1.9	**7.8**	**14.1**	24.7	33.6	55.4	63.3	70.3	**3.5**	**14.6**	24.8	39.7	16.5	20.3	**24.9**	33.5
SIMBA (U)	7.2	15.7	**22.5**	**32.5**	2.4	8.8	**15.3**	**26.5**	1.9	**7.8**	14.0	25.1	30.8	56.7	**64.7**	**71.6**	3.4	14.3	24.7	40.2	16.5	20.3	**24.9**	33.5
SIMBA (U*)	7.2	15.8	**22.5**	32.4	2.4	**8.9**	**15.3**	**26.5**	1.9	**7.8**	**14.1**	25.2	32.1	56.6	64.3	71.2	**3.5**	**14.6**	**24.9**	**40.4**	16.5	**20.4**	**24.9**	**33.6**

For a more comprehensive comparison and to show the competitiveness of SIMBA, we include DynaFLUX (Zhang et al., 2026a) in Table 8, which incorporates more information into its model by observing the node states at each diffusion step and requires access to the underlying diffusion models to run Monte-Carlo simulations to compute the reward during reinforcement learning training.

E.4 Results on Digg using different search parameters

Table 9: Influence spread (%) achieved by SIMBA on Digg using different search parameters.

Setting	Method	Digg (IC)				Digg (LT)				Digg (SIS)			
		1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
# of swaps = 1 # of steps = 50,000	SIMBA (C)	60.6	68.3	74.8	82.3	11.3	22.5	35.2	91.8	16.5	20.5	25.0	33.5
	SIMBA (U)	60.7	68.4	74.8	82.3	11.1	21.7	33.8	91.2	16.5	20.5	24.9	33.6
	SIMBA (U*)	60.4	68.7	76.1	84.0	18.8	82.7	87.4	93.8	16.5	20.7	25.1	33.8
# of swaps = 20 # of steps = 20,000	SIMBA (C)	60.3	66.3	72.3	80.6	12.3	46.4	85.0	90.6	16.5	20.3	24.9	33.5
	SIMBA (U)	60.4	66.5	72.3	80.6	15.4	81.9	85.2	90.9	16.5	20.3	24.9	33.5
	SIMBA (U*)	60.3	66.4	72.5	81.2	18.6	82.4	86.3	91.8	16.5	20.4	24.9	33.6

E.5 Full results on data efficiency

Table 10: Influence spread (%) achieved by all SIMBA variants with varying fractions of training data.

Training data	Method	Cora-ML (IC)				Network Science (IC)				Power Grid (IC)				Jazz (IC)			
		1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
1%	SIMBA (C)	14.2	32.2	39.5	54.4	9.4	24.1	36.9	50.1	8.2	26.3	40.2	57.5	10.1	27.1	36.4	49.4
	SIMBA (U)	16.8	32.2	42.8	57.2	9.7	23.8	37.0	58.0	7.9	25.8	40.2	59.3	12.1	27.5	37.5	49.3
	SIMBA (U*)	16.8	32.4	43.1	57.2	9.7	24.4	37.3	58.0	8.2	26.4	41.1	59.8	12.1	27.2	37.4	49.7
5%	SIMBA (C)	14.1	32.5	39.8	54.4	9.4	24.1	36.8	50.2	8.2	26.4	40.3	57.4	10.4	27.2	36.4	49.7
	SIMBA (U)	16.9	32.5	43.2	56.8	9.7	24.0	36.8	58.0	8.2	25.8	40.4	59.5	12.3	27.5	37.1	50.2
	SIMBA (U*)	16.7	32.7	43.2	56.8	9.7	24.4	37.3	58.2	8.2	26.7	41.1	59.8	11.6	27.2	36.7	50.5
20%	SIMBA (C)	14.2	32.6	39.8	54.5	9.6	24.1	36.9	50.3	8.2	26.6	40.5	57.6	9.3	27.4	36.3	50.3
	SIMBA (U)	16.9	32.4	43.1	57.0	9.7	23.9	37.1	58.0	7.9	25.9	40.6	59.4	11.7	26.6	36.3	50.3
	SIMBA (U*)	17.0	32.9	43.1	57.1	9.5	24.3	37.3	58.2	8.3	26.7	41.4	59.9	12.0	27.6	36.9	50.5
100%	SIMBA (C)	14.2	33.2	39.8	54.7	9.5	24.0	36.6	50.3	8.5	26.7	40.5	57.6	10.4	27.2	36.7	50.0
	SIMBA (U)	16.9	33.3	43.1	57.3	9.7	24.0	36.6	57.8	8.2	26.2	40.4	59.5	12.3	27.6	37.5	51.1
	SIMBA (U*)	17.2	33.6	43.3	57.4	9.6	24.4	37.1	58.1	8.5	26.9	41.3	60.1	11.7	26.7	37.5	50.8

Training data	Method	Cora-ML (LT)				Network Science (LT)				Power Grid (LT)				Jazz (LT)			
		1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
1%	SIMBA (C)	20.9	76.9	84.8	89.3	4.9	21.9	28.5	41.8	7.8	32.5	54.2	77.3	1.3	7.5	72.0	98.1
	SIMBA (U)	18.1	80.8	93.4	98.8	6.9	26.1	42.4	66.8	7.6	30.9	58.0	88.4	2.0	23.0	58.7	100.0
	SIMBA (U*)	18.6	81.2	93.1	98.8	6.9	26.1	42.2	66.8	7.8	32.5	59.3	88.7	2.0	22.9	45.6	100.0
5%	SIMBA (C)	19.9	76.8	84.8	89.4	4.9	21.8	28.5	41.9	7.8	32.2	54.2	77.4	1.3	7.4	70.4	97.9
	SIMBA (U)	19.0	80.7	92.8	98.9	6.9	25.9	41.8	66.2	7.6	30.6	58.3	88.3	2.0	22.7	56.8	100.0
	SIMBA (U*)	18.4	80.5	92.9	98.7	7.1	26.2	41.8	66.4	7.8	32.1	59.4	88.7	2.0	22.5	52.1	100.0
20%	SIMBA (C)	20.7	76.9	84.8	89.4	4.7	21.9	28.6	41.9	7.8	32.3	54.1	77.4	1.4	7.4	70.5	98.3
	SIMBA (U)	19.3	80.7	93.0	98.7	6.8	26.3	42.3	67.0	7.6	30.7	58.2	88.1	2.0	30.4	54.5	100.0
	SIMBA (U*)	18.5	80.7	93.2	98.7	6.7	26.4	42.5	66.7	7.8	32.1	59.5	88.8	2.0	27.0	56.7	100.0
100%	SIMBA (C)	20.7	76.7	84.8	89.4	4.9	22.1	28.6	41.9	7.8	32.5	54.2	77.4	1.3	7.4	71.2	98.6
	SIMBA (U)	18.9	80.0	93.3	98.8	6.8	26.5	41.7	66.4	7.6	31.0	58.4	88.1	2.0	30.4	48.0	100.0
	SIMBA (U*)	18.0	80.3	93.0	98.9	7.0	26.3	41.6	66.6	7.8	32.4	59.5	88.7	2.0	22.5	58.5	100.0

Training data	Method	Cora-ML (SIS)				Network Science (SIS)				Power Grid (SIS)				Jazz (SIS)			
		1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
1%	SIMBA (C)	7.1	15.6	21.5	30.5	2.5	8.7	14.5	23.5	1.8	7.5	13.4	24.1	35.1	55.1	62.9	70.3
	SIMBA (U)	7.0	15.5	21.9	31.6	2.4	9.0	14.6	25.4	1.8	7.4	13.4	24.3	35.2	56.1	64.6	71.7
	SIMBA (U*)	7.1	15.7	22.0	31.6	2.5	9.0	14.7	25.4	1.8	7.4	13.4	24.5	35.0	56.3	64.5	71.5
5%	SIMBA (C)	7.2	15.5	21.8	31.0	2.5	8.8	14.3	23.7	1.8	7.7	13.6	24.4	33.1	55.0	62.9	70.2
	SIMBA (U)	7.0	15.4	22.5	32.3	2.3	9.1	15.1	25.9	1.8	7.5	13.5	24.7	32.8	56.6	64.7	71.4
	SIMBA (U*)	7.2	15.5	22.6	32.3	2.3	9.1	15.2	25.9	1.8	7.6	13.6	24.9	31.8	56.8	64.5	71.4
20%	SIMBA (C)	7.2	15.8	21.8	31.0	2.5	8.9	14.6	23.8	1.8	7.7	14.0	24.6	33.8	54.8	63.4	70.2
	SIMBA (U)	7.1	15.8	22.4	32.4	2.4	8.8	15.2	26.0	1.8	7.6	13.8	25.0	28.9	56.6	64.9	71.2
	SIMBA (U*)	7.2	15.9	22.4	32.5	2.4	8.8	15.1	26.0	1.9	7.8	14.0	25.1	29.9	56.9	64.6	71.5
100%	SIMBA (C)	7.2	15.7	21.9	31.1	2.5	8.8	14.6	23.8	1.9	7.8	14.1	24.7	33.6	55.4	63.3	70.3
	SIMBA (U)	7.2	15.7	22.5	32.5	2.4	8.8	15.3	26.5	1.9	7.8	14.0	25.1	30.8	56.7	64.7	71.6
	SIMBA (U*)	7.2	15.8	22.5	32.4	2.4	8.9	15.3	26.5	1.9	7.8	14.1	25.2	32.1	56.6	64.3	71.2

E.6 FULL RESULTS ON THE EFFECT OF NODE EMBEDDING INITIALIZATION STRATEGY ON SEED SET QUALITY

Table 11: Influence spread (%) achieved by all SIMBA variants with different node embedding initialization strategies and post-training embedding update rules.

		Cora-ML (IC)				Network Science (IC)				Power Grid (IC)				Jazz (IC)			
Setting	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
Random No update	SIMBA (C)	14.0	32.3	39.1	53.9	9.2	23.9	36.7	49.6	8.1	26.2	40.0	56.8	9.8	26.9	35.5	49.3
	SIMBA (U)	16.4	32.4	42.4	56.3	9.4	23.7	36.5	56.9	7.9	25.7	39.8	58.7	10.4	25.3	36.1	50.5
	SIMBA (U*)	16.5	32.8	42.3	56.1	9.5	23.9	36.6	57.1	8.0	26.2	40.6	59.0	9.9	25.8	36.7	50.6
Random Weighted avg	SIMBA (C)	14.0	32.2	39.4	53.8	9.2	23.9	36.5	49.7	8.2	26.2	39.8	56.6	10.3	26.5	35.9	49.3
	SIMBA (U)	14.8	32.4	38.9	53.0	9.2	23.7	36.0	49.6	7.9	25.4	38.5	55.0	10.6	26.8	36.3	49.4
	SIMBA (U*)	14.8	32.5	39.3	53.8	9.2	24.0	36.6	49.7	8.3	26.4	40.0	56.8	11.1	26.9	35.6	49.1
Random Most similar	SIMBA (C)	13.9	32.5	39.5	54.0	9.3	24.0	36.6	49.7	8.2	26.2	39.8	56.7	8.7	26.6	35.5	50.0
	SIMBA (U)	16.7	32.5	42.7	56.2	9.3	24.0	36.6	56.7	7.8	25.5	39.8	58.8	9.7	26.8	36.6	50.5
	SIMBA (U*)	16.4	32.8	42.8	56.3	9.4	24.4	36.8	56.4	8.2	26.3	40.5	59.1	9.7	26.9	36.4	50.1
Random Global mean	SIMBA (C)	14.2	32.2	39.3	53.8	9.3	23.9	36.5	49.8	8.1	26.2	39.8	56.4	10.1	26.3	36.4	49.1
	SIMBA (U)	14.4	32.3	38.6	53.3	9.3	23.7	36.1	49.4	8.0	25.3	38.8	54.7	10.4	26.5	35.9	49.4
	SIMBA (U*)	14.6	32.6	39.4	53.9	9.4	24.0	36.5	49.8	8.2	26.3	40.0	56.5	9.9	26.8	36.1	49.3
Anchored No update	SIMBA (C)	14.2	33.2	39.8	54.7	9.5	24.0	36.6	50.3	8.5	26.7	40.5	57.6	10.4	27.2	36.7	50.0
	SIMBA (U)	16.9	33.3	43.1	57.3	9.7	24.0	36.6	57.8	8.2	26.2	40.4	59.5	12.3	27.6	37.5	51.1
	SIMBA (U*)	17.2	33.6	43.3	57.4	9.6	24.4	37.1	58.1	8.5	26.9	41.3	60.1	11.7	26.7	37.5	50.8

		Cora-ML (LT)				Network Science (LT)				Power Grid (LT)				Jazz (LT)			
Setting	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
Random No update	SIMBA (C)	16.8	75.5	84.3	89.3	4.7	21.0	28.6	41.9	7.3	30.8	53.4	76.8	1.4	6.9	69.8	88.9
	SIMBA (U)	15.3	79.1	90.9	97.6	5.6	22.0	35.2	59.4	6.6	26.7	52.7	83.1	1.8	7.7	24.0	99.5
	SIMBA (U*)	15.0	79.1	91.1	97.6	5.8	21.9	35.0	59.3	6.8	28.2	54.2	83.6	1.8	7.7	24.1	99.0
Random Weighted avg	SIMBA (C)	17.3	75.5	84.3	89.3	4.7	20.9	28.6	41.9	7.3	30.9	53.5	77.1	1.3	7.1	67.9	85.6
	SIMBA (U)	16.9	74.1	84.0	89.3	4.7	21.0	29.8	43.0	7.0	29.2	51.3	75.8	1.9	7.1	68.6	94.9
	SIMBA (U*)	17.6	75.6	84.4	89.2	4.7	20.9	28.8	42.0	7.3	30.9	53.4	77.1	1.9	7.1	67.7	94.8
Random Most similar	SIMBA (C)	16.3	74.9	84.3	89.3	4.8	20.9	28.5	41.9	7.3	30.7	53.5	77.1	1.4	7.0	67.9	88.9
	SIMBA (U)	15.3	78.0	90.9	97.4	6.6	22.0	37.6	61.7	7.0	27.3	54.0	85.0	2.0	13.1	24.5	98.8
	SIMBA (U*)	15.6	78.8	90.5	97.6	6.5	22.3	37.6	61.4	7.1	28.7	54.8	85.9	2.0	13.1	25.0	99.5
Random Global mean	SIMBA (C)	16.9	74.7	84.3	89.3	4.8	21.1	28.5	41.8	7.3	31.1	53.3	76.8	1.3	7.1	68.8	86.2
	SIMBA (U)	16.1	73.7	84.4	89.6	4.9	21.1	28.9	42.3	7.1	29.8	51.1	75.6	1.9	7.1	66.4	99.5
	SIMBA (U*)	16.9	74.9	84.4	89.3	4.9	21.1	28.5	41.9	7.3	31.0	53.2	76.8	1.9	7.1	64.9	99.5
Anchored No update	SIMBA (C)	20.7	76.7	84.8	89.4	4.9	22.1	28.6	41.9	7.8	32.5	54.2	77.4	1.3	7.4	71.2	98.6
	SIMBA (U)	18.9	80.0	93.3	98.8	6.8	26.5	41.7	66.4	7.6	31.0	58.4	88.1	2.0	30.4	48.0	100.0
	SIMBA (U*)	18.0	80.3	93.0	98.9	7.0	26.3	41.6	66.6	7.8	32.4	59.5	88.7	2.0	22.5	58.5	100.0

		Cora-ML (SIS)				Network Science (SIS)				Power Grid (SIS)				Jazz (SIS)			
Setting	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
Random No update	SIMBA (C)	7.3	16.0	22.0	31.3	2.9	8.9	14.6	23.6	1.9	7.8	14.0	24.5	33.1	52.6	61.3	69.4
	SIMBA (U)	6.8	15.5	22.5	32.9	2.1	8.1	14.5	25.8	1.6	7.2	13.3	24.4	31.6	54.6	62.8	70.8
	SIMBA (U*)	6.9	15.6	22.6	32.9	2.0	8.2	14.5	25.7	1.7	7.3	13.4	24.5	32.0	54.1	63.3	70.8
Random Weighted avg	SIMBA (C)	7.3	16.0	22.0	31.4	2.9	8.9	14.6	23.6	1.9	7.8	14.0	24.5	32.9	52.4	61.6	69.4
	SIMBA (U)	7.2	15.8	21.9	31.2	2.9	8.9	14.4	23.6	1.8	7.7	13.8	24.2	32.3	54.0	61.6	69.9
	SIMBA (U*)	7.3	16.0	22.0	31.4	2.9	8.9	14.6	23.6	1.9	7.8	14.0	24.5	32.8	54.0	61.8	69.8
Random Most similar	SIMBA (C)	7.2	16.0	21.9	31.3	2.9	8.9	14.6	23.7	1.9	7.8	14.0	24.5	34.5	53.0	61.7	69.1
	SIMBA (U)	7.1	15.7	22.6	32.6	2.7	9.0	15.3	26.2	1.8	7.4	13.6	24.4	31.1	53.2	62.7	70.0
	SIMBA (U*)	7.2	15.9	22.6	32.6	2.7	9.0	15.3	26.3	1.9	7.6	13.7	24.6	29.9	53.5	62.9	70.3
Random Global mean	SIMBA (C)	7.3	16.1	22.0	31.3	2.9	8.9	14.6	23.6	1.9	7.8	14.0	24.5	32.6	53.0	61.6	68.7
	SIMBA (U)	7.2	15.8	21.8	31.2	2.9	8.8	14.5	23.6	1.8	7.7	13.8	24.2	32.7	54.2	61.5	68.5
	SIMBA (U*)	7.3	16.0	22.0	31.3	2.9	8.9	14.5	23.6	1.9	7.8	14.0	24.5	33.5	54.0	61.6	68.4
Anchored No update	SIMBA (C)	7.2	15.7	21.9	31.1	2.5	8.8	14.6	23.8	1.9	7.8	14.1	24.7	33.6	55.4	63.3	70.3
	SIMBA (U)	7.2	15.7	22.5	32.5	2.4	8.8	15.3	26.5	1.9	7.8	14.0	25.1	30.8	56.7	64.7	71.6
	SIMBA (U*)	7.2	15.8	22.5	32.4	2.4	8.9	15.3	26.5	1.9	7.8	14.1	25.2	32.1	56.6	64.3	71.2

E.7 Results on applying embedding update rules to uniformly anchored node embeddings

Table 12: Influence spread (%) achieved by SIMBA with uniformly anchored node embeddings and different post-training embedding update rules.

		Cora-ML (IC)				Network Science (IC)				Power Grid (IC)				Jazz (IC)			
Setting	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
Anchored Weighted avg	SIMBA (C)	14.1	32.9	39.9	54.7	9.5	23.6	36.8	50.3	8.4	26.7	40.4	57.7	11.1	27.1	37.2	50.1
	SIMBA (U)	16.9	32.9	43.2	57.6	9.7	23.8	36.8	58.0	8.2	26.0	40.6	59.7	12.1	26.5	37.1	50.7
	SIMBA (U*)	16.9	33.2	43.3	57.7	9.8	24.1	37.0	58.3	8.4	26.9	41.3	60.0	11.9	26.6	37.4	50.4
Anchored Most similar	SIMBA (C)	14.3	33.2	39.8	54.5	9.5	24.0	36.9	50.1	8.4	26.6	40.4	57.5	10.5	27.8	36.6	50.2
	SIMBA (U)	16.2	33.1	43.2	57.1	9.6	23.8	36.9	58.1	8.2	25.9	40.3	59.4	12.1	27.4	37.4	50.0
	SIMBA (U*)	16.5	33.6	43.3	57.1	9.7	24.2	37.2	58.1	8.4	26.8	41.2	59.9	12.0	27.7	36.7	50.2
Anchored Global mean	SIMBA (C)	14.2	33.2	39.9	54.6	9.5	24.1	36.8	50.3	8.5	26.6	40.4	57.5	10.1	27.0	36.8	50.0
	SIMBA (U)	17.2	33.1	43.4	56.9	9.6	24.1	37.1	58.1	8.1	26.1	40.5	59.6	11.9	26.8	37.5	50.3
	SIMBA (U*)	16.9	33.5	43.2	57.0	9.5	24.3	37.4	58.3	8.4	26.8	41.4	60.0	11.9	26.7	37.8	50.2
Anchored No update	SIMBA (C)	14.2	33.2	39.8	54.7	9.5	24.0	36.6	50.3	8.5	26.7	40.5	57.6	10.4	27.2	36.7	50.0
	SIMBA (U)	16.9	33.3	43.1	57.3	9.7	24.0	36.6	57.8	8.2	26.2	40.4	59.5	12.3	27.6	37.5	51.1
	SIMBA (U*)	17.2	33.6	43.3	57.4	9.6	24.4	37.1	58.1	8.5	26.9	41.3	60.1	11.7	26.7	37.5	50.8

		Cora-ML (LT)				Network Science (LT)				Power Grid (LT)				Jazz (LT)			
Setting	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
Anchored Weighted avg	SIMBA (C)	20.7	76.9	84.8	89.4	4.8	22.0	28.6	41.9	7.8	32.4	54.2	77.5	1.3	7.4	70.5	98.3
	SIMBA (U)	17.9	80.4	93.1	98.8	7.1	26.2	42.2	65.7	7.5	30.9	58.1	88.4	2.0	30.4	52.4	100.0
	SIMBA (U*)	19.8	81.3	93.4	98.7	7.0	26.4	42.2	66.0	7.8	32.5	59.5	88.7	2.0	26.8	60.1	100.0
Anchored Most similar	SIMBA (C)	20.6	76.7	84.8	89.4	4.6	21.7	28.6	41.9	7.8	32.4	54.3	77.4	1.4	7.5	70.6	98.2
	SIMBA (U)	17.2	80.4	92.7	98.8	7.1	25.4	42.1	66.9	7.6	30.7	58.3	88.3	1.9	30.4	50.7	100.0
	SIMBA (U*)	18.1	80.1	92.5	98.9	7.2	25.6	41.9	66.9	7.9	32.1	59.2	88.5	1.9	26.9	59.5	100.0
Anchored Global mean	SIMBA (C)	20.5	76.8	84.7	89.4	4.8	21.9	28.6	41.9	7.8	32.5	54.3	77.4	1.3	7.4	71.5	98.1
	SIMBA (U)	18.3	80.2	93.2	98.8	6.8	25.8	42.4	66.6	7.6	30.7	58.0	88.4	2.0	30.6	54.6	100.0
	SIMBA (U*)	18.7	80.2	93.5	98.8	7.2	26.2	42.3	66.8	7.8	32.5	59.0	88.8	2.0	21.8	59.3	100.0
Anchored No update	SIMBA (C)	20.7	76.7	84.8	89.4	4.9	22.1	28.6	41.9	7.8	32.5	54.2	77.4	1.3	7.4	71.2	98.6
	SIMBA (U)	18.9	80.0	93.3	98.8	6.8	26.5	41.7	66.4	7.6	31.0	58.4	88.1	2.0	30.4	48.0	100.0
	SIMBA (U*)	18.0	80.3	93.0	98.9	7.0	26.3	41.6	66.6	7.8	32.4	59.5	88.7	2.0	22.5	58.5	100.0

		Cora-ML (SIS)				Network Science (SIS)				Power Grid (SIS)				Jazz (SIS)			
Setting	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
Anchored Weighted avg	SIMBA (C)	7.3	15.8	21.9	31.0	2.5	8.8	14.6	23.8	1.9	7.8	14.1	24.7	33.7	55.0	63.0	70.5
	SIMBA (U)	7.1	15.8	22.2	32.4	2.4	8.8	15.4	26.4	1.9	7.7	13.9	25.2	28.4	56.2	64.2	71.5
	SIMBA (U*)	7.2	15.8	22.4	32.4	2.4	8.9	15.5	26.4	1.9	7.8	14.1	25.3	31.1	55.9	64.6	71.4
Anchored Most similar	SIMBA (C)	7.3	15.8	21.9	31.1	2.6	8.7	14.7	23.8	1.9	7.8	14.1	24.7	33.3	55.1	62.7	70.1
	SIMBA (U)	7.1	15.7	22.4	32.5	2.5	8.9	15.4	26.0	1.9	7.7	13.9	25.1	29.9	56.1	64.5	71.5
	SIMBA (U*)	7.2	15.8	22.3	32.4	2.5	8.8	15.3	26.0	1.9	7.9	14.1	25.2	32.1	55.9	64.7	71.1
Anchored Global mean	SIMBA (C)	7.3	15.8	21.9	31.1	2.5	8.8	14.6	23.8	1.9	7.8	14.1	24.7	34.7	55.7	63.3	70.5
	SIMBA (U)	7.1	15.8	22.3	32.5	2.3	8.9	15.3	26.2	1.9	7.8	14.0	25.1	29.9	56.3	64.4	71.4
	SIMBA (U*)	7.2	15.8	22.3	32.6	2.3	8.9	15.4	26.1	1.9	7.8	14.1	25.3	30.9	56.3	64.6	71.4
Anchored No update	SIMBA (C)	7.2	15.7	21.9	31.1	2.5	8.8	14.6	23.8	1.9	7.8	14.1	24.7	33.6	55.4	63.3	70.3
	SIMBA (U)	7.2	15.7	22.5	32.5	2.4	8.8	15.3	26.5	1.9	7.8	14.0	25.1	30.8	56.7	64.7	71.6
	SIMBA (U*)	7.2	15.8	22.5	32.4	2.4	8.9	15.3	26.5	1.9	7.8	14.1	25.2	32.1	56.6	64.3	71.2

E.8 INFLUENCE MAXIMIZATION PERFORMANCE OF SIMBA WITH DIFFERENT NODE EMBEDDING DIMENSIONS

Table 13: Influence spread (%) achieved by SIMBA with 2D uniformly anchored node embeddings.

		Cora-ML (IC)				Network Science (IC)				Power Grid (IC)				Jazz (IC)			
Embedding	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
$d = 2$	SIMBA (C)	13.9	32.8	39.4	54.5	9.2	24.0	36.9	50.2	8.3	26.3	40.0	57.3	11.0	27.0	36.3	49.5
	SIMBA (U)	16.9	32.7	42.6	56.9	9.4	24.1	37.2	57.6	8.0	25.9	40.2	59.3	11.2	27.0	36.8	51.0
	SIMBA (U*)	16.9	33.2	42.6	56.9	9.4	24.3	37.4	57.8	8.3	26.7	40.9	59.8	12.0	27.8	36.9	50.9
$d = 8$	SIMBA (C)	14.2	33.2	39.8	54.7	9.5	24.0	36.6	50.3	8.5	26.7	40.5	57.6	10.4	27.2	36.7	50.0
	SIMBA (U)	16.9	33.3	43.1	57.3	9.7	24.0	36.6	57.8	8.2	26.2	40.4	59.5	12.3	27.6	37.5	51.1
	SIMBA (U*)	17.2	33.6	43.3	57.4	9.6	24.4	37.1	58.1	8.5	26.9	41.3	60.1	11.7	26.7	37.5	50.8

		Cora-ML (LT)				Network Science (LT)				Power Grid (LT)				Jazz (LT)			
Embedding	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
$d = 2$	SIMBA (C)	19.3	75.1	84.8	89.3	4.8	21.8	28.6	41.9	7.7	32.3	54.3	77.4	1.3	7.5	71.9	95.4
	SIMBA (U)	17.8	77.6	92.7	98.5	6.9	26.0	41.3	65.7	7.5	30.7	58.6	88.4	1.8	30.4	59.9	99.8
	SIMBA (U*)	18.5	77.2	92.8	98.5	7.0	26.0	41.6	65.5	7.8	32.2	59.5	88.7	1.8	26.0	60.9	99.9
$d = 8$	SIMBA (C)	20.7	76.7	84.8	89.4	4.9	22.1	28.6	41.9	7.8	32.5	54.2	77.4	1.3	7.4	71.2	98.6
	SIMBA (U)	18.9	80.0	93.3	98.8	6.8	26.5	41.7	66.4	7.6	31.0	58.4	88.1	2.0	30.4	48.0	100.0
	SIMBA (U*)	18.0	80.3	93.0	98.9	7.0	26.3	41.6	66.6	7.8	32.4	59.5	88.7	2.0	22.5	58.5	100.0

		Cora-ML (SIS)				Network Science (SIS)				Power Grid (SIS)				Jazz (SIS)			
Embedding	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
$d = 2$	SIMBA (C)	7.3	15.8	21.9	31.0	2.6	8.8	14.7	23.8	1.9	7.9	14.1	24.7	34.4	54.6	63.3	69.9
	SIMBA (U)	7.2	15.8	22.4	32.3	2.5	8.7	15.4	26.2	1.8	7.7	14.0	25.1	34.7	56.2	64.5	70.6
	SIMBA (U*)	7.3	16.0	22.3	32.3	2.5	8.8	15.3	26.2	1.9	7.9	14.2	25.2	34.5	55.9	64.4	71.1
$d = 8$	SIMBA (C)	7.2	15.7	21.9	31.1	2.5	8.8	14.6	23.8	1.9	7.8	14.1	24.7	33.6	55.4	63.3	70.3
	SIMBA (U)	7.2	15.7	22.5	32.5	2.4	8.8	15.3	26.5	1.9	7.8	14.0	25.1	30.8	56.7	64.7	71.6
	SIMBA (U*)	7.2	15.8	22.5	32.4	2.4	8.9	15.3	26.5	1.9	7.8	14.1	25.2	32.1	56.6	64.3	71.2

E.9 ARCHITECTURAL ROBUSTNESS

Table 14: Influence spread (%) achieved by SIMBA with binary seed vector input.

		Cora-ML (IC)				Network Science (IC)				Power Grid (IC)				Jazz (IC)			
Embedding	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
Binary	SIMBA (C)	13.9	31.7	39.7	54.3	9.3	24.0	36.9	50.2	8.1	26.3	40.0	57.4	10.0	27.4	36.2	50.3
	SIMBA (U)	16.9	31.8	43.2	56.1	9.6	24.1	37.3	57.9	8.2	26.0	40.2	59.2	11.1	**27.7**	**37.9**	50.5
	SIMBA (U*)	17.0	31.8	42.9	56.1	9.5	24.3	**37.4**	**58.3**	8.3	26.6	41.1	59.8	11.7	27.6	38.2	50.3
$d = 8$	SIMBA (C)	14.2	33.2	39.8	54.7	9.5	24.0	36.6	50.3	**8.5**	26.7	40.5	57.6	10.4	27.2	36.7	50.0
	SIMBA (U)	16.9	33.3	43.1	57.3	**9.7**	24.0	36.6	57.8	8.2	26.2	40.4	59.5	**12.3**	27.6	37.5	**51.1**
	SIMBA (U*)	**17.2**	**33.6**	**43.3**	**57.4**	9.6	**24.4**	37.1	58.1	**8.5**	**26.9**	**41.3**	**60.1**	11.7	26.7	37.5	50.8

		Cora-ML (LT)				Network Science (LT)				Power Grid (LT)				Jazz (LT)			
Embedding	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
Binary	SIMBA (C)	20.3	76.7	84.8	89.4	5.2	21.6	28.6	41.9	7.6	32.0	54.3	77.5	1.4	7.4	**71.8**	98.4
	SIMBA (U)	17.7	79.9	92.8	98.6	6.8	26.0	41.5	64.6	7.3	30.2	58.2	88.4	**2.0**	**30.5**	49.5	99.9
	SIMBA (U*)	18.2	80.2	92.8	98.7	**7.0**	26.1	41.5	64.5	7.7	31.7	59.1	88.4	**2.0**	26.4	56.9	99.9
$d = 8$	SIMBA (C)	**20.7**	76.7	84.8	89.4	4.9	22.1	28.6	41.9	**7.8**	**32.5**	54.2	77.4	1.3	7.4	71.2	98.6
	SIMBA (U)	18.9	80.0	**93.3**	98.8	6.8	**26.5**	**41.7**	66.4	7.6	31.0	58.4	88.1	**2.0**	30.4	48.0	**100.0**
	SIMBA (U*)	18.0	**80.3**	93.0	**98.9**	**7.0**	26.3	41.6	**66.6**	**7.8**	32.4	**59.5**	**88.7**	**2.0**	22.5	58.5	**100.0**

		Cora-ML (SIS)				Network Science (SIS)				Power Grid (SIS)				Jazz (SIS)			
Embedding	Method	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
Binary	SIMBA (C)	**7.4**	**15.8**	21.9	31.2	**2.5**	**8.9**	14.7	23.8	**1.9**	**7.8**	**14.1**	24.7	35.1	54.8	63.4	70.5
	SIMBA (U)	7.2	15.7	22.5	**32.6**	**2.5**	8.6	14.8	26.4	**1.9**	7.7	14.0	25.1	**35.3**	56.4	64.6	71.5
	SIMBA (U*)	7.2	**15.8**	**22.6**	32.4	**2.5**	8.5	14.8	26.3	**1.9**	**7.8**	**14.1**	**25.3**	**35.3**	**56.7**	64.6	71.5
$d = 8$	SIMBA (C)	7.2	15.7	21.9	31.1	**2.5**	8.8	14.6	23.8	**1.9**	**7.8**	**14.1**	24.7	33.6	55.4	63.3	70.3
	SIMBA (U)	7.2	15.7	22.5	32.5	2.4	8.8	**15.3**	**26.5**	**1.9**	**7.8**	14.0	25.1	30.8	**56.7**	**64.7**	**71.6**
	SIMBA (U*)	7.2	**15.8**	22.5	32.4	2.4	**8.9**	**15.3**	**26.5**	**1.9**	**7.8**	**14.1**	25.2	32.1	56.6	64.3	71.2

Finally, we evaluate an extremely lightweight variant where each node is represented using only its binary seed indicator. As reported in Table 14, although learnable embeddings boost IM performance, this simplified representation remains surprisingly competitive, suggesting that most of the predictive power arises from topology-aware and diffusion-dependent message passing rather than from high-dimensional node embeddings. This provides further evidence to support our core design philosophy that a simple, lightweight neural surrogate is sufficient to guide the search for high-quality seed sets.