DIFFIM: APPENDIX

A Proofs

A.1 Proof of Theorem 1

Proof. We consider an NP-hard problem called minimum k-union (Vinterbo 2002), and we shall show that given each instance of minimum k-union, we can construct an instance of influence minimization, such that if the instance of influence minimization is our problem is solved, then the original instance of minimum k-union is solved. Given a collection of m set $S_1, S_2, \ldots, S_m$ and a positive integer k, the minimum k-union problem aims to find k sets $(S_{i_1}, S_{i_2}, \ldots, S_{i_k})$ such that $|\bigcup_j S_{i_j}|$ is minimized. Let

$$X = \{x_1, x_2, \ldots, x_n\} = \bigcup_{j \in [m]} S_j.$$

Given any instance of minimum k-union, we construct the following instance of influence minimization: we have a single seed node $S = \{v_{init}\}$, m nodes $v_j^{(S)}$ for $j \in [m]$, and n nodes $v_i^{(X)}$ for $i \in [n]$. We have m edges from v_{init} to each $v_j^{(S)}$, and edge from $v_j^{(S)}$ to $v_i^{(X)}$ if and only if $x_i \in S_j$, for each (i, j) pair. Also, each edge e has an activation probability $p(e) = 1$. Now, it is easy to see that the original instance of minimum k-union is equivalent to finding $m - k$ edges between v_{init} and $v_j^{(S)}$ such that removing those $m-k$ edges will minimize the expected number of activated nodes in the whole process (i.e., the objective in influence minimization). Hence, it suffices to show that you cannot do better by removing other edges. Indeed, if you remove an edge from $v_j^{(S)}$ to $v_i^{(X)}$, then replacing it by removing the edge from v_{init} to $v_j^{(S)}$ will give at least the same, or better, minimization performance. □

A.2 Proof of Lemma 1

Proof. We have

$$\frac{\partial \operatorname{GNN}_\theta(v; G, \tilde{p}_{\tilde{r}}, S)}{\partial \tilde{r}} = \frac{\partial \operatorname{GNN}_\theta(v; G, \tilde{p}_{\tilde{r}}, S)}{\partial \tilde{p}_{\tilde{r}}} \frac{\partial \tilde{p}_{\tilde{r}}}{\partial \tilde{r}},$$

where

$$\frac{\partial \tilde{p}_{\tilde{r}}(v, u)}{\partial \tilde{r}(v', u')} = p(v, u)\mathbb{1}(v = v' \wedge u = u').$$

As mentioned in Sec. 5.2, p is used as edge weights in the input graph of GNN, and thus $\frac{\partial \operatorname{GNN}_\theta(v;G,\tilde{p}_{\tilde{r}},S)}{\partial \tilde{p}_{\tilde{r}}}$ is also well-defined. Hence, $\frac{\partial \operatorname{GNN}_\theta(v;G,\tilde{p}_{\tilde{r}},S)}{\partial \tilde{r}}$ is well-defined. □

A.3 Proof of the meaningfulness of the loss Function

Lemma 2. *Given a test instance $\mathcal{T} = (G = (V, E), p, S, b)$, let*

$$\sigma^* \coloneqq \min_{\mathcal{E} \subseteq E, |\mathcal{E}| = b} \sigma(S; G_{\setminus \mathcal{E}}, p_{\setminus \mathcal{E}}).$$

Assume that (1) $\operatorname{GNN}_\theta$ is well trained, i.e.,

$$|\operatorname{GNN}_\theta(v; G, p, S) - \pi(v; G, p, S)| \le \epsilon, \forall v$$

for some $\epsilon > 0$ and (2) α and β are sufficiently large, then each $r^* \in \arg\min_{\tilde{r}} \mathcal{L}_O(\tilde{r})$ satisfies that (1) r^* is discrete, i.e.,

$$r^*(e) \in \{0, 1\}, \forall e \in E,$$

(2) r^* satisfies the budget constraint, i.e.,

$$\sum_{e \in E} r^*(e) = |E| - b,$$

and (3) r^* is a good solution, i.e.,

$$\sigma(S; G_{\setminus \mathcal{E}^*}, p_{\setminus \mathcal{E}^*}) - \sigma^* \le 2|V|\epsilon,$$

where

$$\mathcal{E}^* = \mathcal{E}^*(r^*) = \{e \in E \colon r^*(e) = 0\}.$$

Proof. Since α and β are sufficiently large and there exists $\tilde{r} : E \to [0, 1]$ such that (1) $\mathcal{L}_{\text{budget}}$ is minimized, i.e.,

$$\mathcal{L}_{\text{budget}}(\tilde{r}) = 0,$$

and (2) $\mathcal{L}_{\text{certainty}}$ is minimized, i.e.,

$$\mathcal{L}_{\text{certainty}}(\tilde{r}) = 0,$$

r^* must satisfy that $\mathcal{L}_{\text{budget}}(r^*) = \mathcal{L}_{\text{certainty}}(r^*) = 0$, which is equivalent to $r^*(e) \in \{0, 1\}, \forall e \in E$ and $\sum_{e \in E} r^*(e) = |E| - b$. In fact, all the discrete and budget-satisfying probabilistic decisions $\tilde{r}$ satisfy that $\mathcal{L}_{\text{budget}}(\tilde{r}) = \mathcal{L}_{\text{certainty}}(\tilde{r}) = 0$, and we have

$$r^* \in \arg \min_{\tilde{r}: E \to \{0,1\}, \sum_{e \in E} \tilde{r}(e) = |E| - b} \mathcal{L}_{\text{obj}},$$

which is equivalent to

$$r^* \in \arg \min_{\tilde{r}: E \to \{0,1\}, \sum_{e \in E} \tilde{r}(e) = |E| - b} \sum_{v \in V} \operatorname{GNN}_\theta(v; G, \tilde{p}_{\tilde{r}}, S).$$

Define $\mathcal{E}_r \coloneqq \{e \in E \colon r(e)\}$ for each $r : E \to \{0, 1\}$, and define

$$\tilde{r}_{\mathcal{E}}(e) = \mathbb{1}(e \notin \mathcal{E}), \forall e \in E.$$

Let

$$\mathcal{E}_{min} \in \arg \min_{\mathcal{E} \subseteq E, |\mathcal{E}| = b} \sigma(S; G_{\setminus \mathcal{E}}, p_{\setminus \mathcal{E}}),$$

for each $v \in V$, we have

$$\begin{aligned} \pi(v; G_{\setminus \mathcal{E}_{r^*}}, p_{\setminus \mathcal{E}_{r^*}}, S) &\le \operatorname{GNN}_\theta(v; G, \tilde{p}_{r^*}, S) + \epsilon \\ &\le \operatorname{GNN}_\theta(v; G, \tilde{p}_{\tilde{r}_{\mathcal{E}_{min}}}, S) + \epsilon \\ &\le \pi(v; G_{\setminus \mathcal{E}_{min}}, p_{\setminus \mathcal{E}_{min}}, S) + 2\epsilon. \end{aligned}$$

Taking the summation over $v \in V$ completes the proof. □

Remark 1. *In practice, however, using too large α and β would make $\mathcal{L}_{budget}$ and $\mathcal{L}_{certainty}$ dominant and thus impair the optimization performance w.r.t. the main objective $\mathcal{L}_{obj}$. See Sec. 6.6 for the ablation studies on α and β.*

B Extension to other spread models

We performed additional experiments under different realistic influence spread models: the linear threshold (LT) model (Kempe, Kleinberg, and Tardos 2003) and the general Markov chain susceptible-infected-recovered (G-SIR) model (Yi et al. 2022).

Model definition: Below, we describe the definitions of the LT model and the G-SIR model.

Definition 3 (Linear threshold (LT) model)**.** *Given (1) a graph $G = (V, E)$ and (2) a seed set $S \subseteq V$, $\mathrm{LT}(G, S)$ is a stochastic process as follows:*

- ***Initialization:*** *At time step $t = 0$, each seed node $v_S \in S$ is activated, and each non-seed node remains inactive. For each $v \in V$, activation threshold p_v, drawn from continuous uniform distribution $U(0, 1)$, is assigned to E.*
- ***Diffusion steps:*** *At each step $t \geq 1$, each inactive node v until the previous step is newly activated when the ratio of its active in-neighbors exceeds the activation threshold p_v. Each activated node remains active for the whole process, and the process terminates when no node is activated in the previous step.*

Definition 4 (General Markov chain susceptible-infected-recovered (G-SIR) model)**.** *Given (1) a graph $G = (V, E)$, (2) activation probabilities $p : E \to [0, 1]$, (3) a recovery probability r, and (4) a seed set $S \subseteq V$, $\text{G-SIR}(G, S)$ is a stochastic process as follows:*

- ***Initialization:*** *At time step $t = 0$, each seed node $v_S \in S$ is activated, and each non-seed node remains inactive.*
- ***Diffusion steps:*** *At each step $t \geq 1$, each active node v at the previous step becomes inactive again with the recovery probability r and activates each of its inactive out-neighbor u with activation probability $p(v, u)$. Each recovered node remains inactive for the whole process. As long as each activated node stays activated, other non-recovered nodes can be activated during the process. The process terminates when no node is activated in the previous step.*

In the G-SIR model, each edge has its individual (and possibly different) propagation probability, while in the original SIR model, the propagation probability is the same for all edges. In our experiments, we set the recovery probability $r = 0.5$.

Extensions: Each considered method (see Sec. 6.1) was straightforwardly extended for the two additional influence spread models. For example, for extending DIFFIM, we modified the influence-estimation component (i.e., the surrogate-model GNN; see Alg. 3) to use the corresponding influence spread model.

Experimental results: As in Sec. 6.2, we measured the average reduced ratio of influence and the average running time across all the test seed sets. We used the same process described in Sec. 6.1 to generate the training and test data for the LT and G-SIR models. As shown in Fig. 4, the results followed a similar trend presented in Sec. 6.2. That is, DIFFIM++ was significantly faster than the most effective baseline, while achieving a similar reduced ratio. For WC, RIS-ϵ ran out of time for every $\epsilon \in \{0.2, 0.4, 0.6\}$, leaving no reasonably comparable baselines.

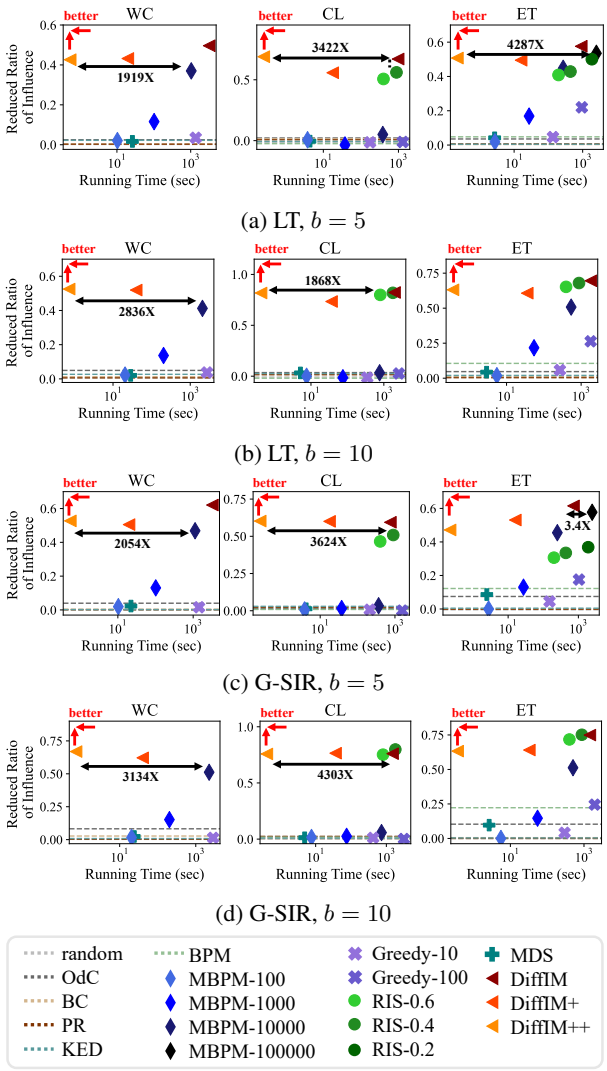


Figure 4: The effectiveness (the reduced ratio of influence) and running time of each method, with budget $b \in \{5, 10\}$ on LT and G-SIR models. We compared the running time of the best baseline to one of our methods with the most similar reduced ratio. Except for (d) on CL and ET, all DIFFIM versions were Pareto-optimal, i.e., no baseline is faster *and* more effective than any version.

C Pseudo-codes of algorithms

We provide the pseudo-codes of the naive incremental greedy algorithm (Alg. 2; see Sec. 5.1) and GNN training (Alg. 3; see Sec. 5.2).

D Additional related work

Here, we provide more details of the related work on influence minimization.

Node removal: The following works considered influence minimization with node removal. Wang et al. (2013) employed an incremental greedy algorithm under the IC model. Chang, Yeh, and Chuang (2016) focused on the outbreak phe-

Algorithm 2: An incremental greedy algorithm

Input: (1) $G = (V, E)$: an input graph
(2) $p : E \to [0, 1]$: activation probabilities
(3) $S \subseteq V$: a seed set
(4) b: an edge-removal budget
Output: $\mathcal{E} \subseteq E$: a set of edges chosen to be removed

$\mathcal{E} \leftarrow \emptyset$ ▷ Initialization
for $i = 1, 2, \ldots, b$ **do**
  $e = \arg\min_{e \in E \setminus \mathcal{E}} \sigma(S; G_{\setminus \mathcal{E}}, p_{\setminus \mathcal{E}})$
  $\mathcal{E} \leftarrow \mathcal{E} \cup \{e\}$ ▷ Incremental update
return $\mathcal{E}$

Algorithm 3: GNN training for influence estimation (executed once, applicable to all future unseen cases)

Input: (1) $G = (V, E)$: an input graph
(2) $p : E \to [0, 1]$: activation probabilities
(3) $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$: a set of seed sets
(4) GNN_θ: a GNN to be trained
Output: GNN_θ: a trained GNN

Obtain $\tilde{\pi}(v; G, S_i, p)$ by MC, for each $v \in V$ and $S_i \in \mathcal{S}$
Update θ w.r.t. $\mathcal{L}(\mathrm{GNN}_\theta; \tilde{\pi}, \mathcal{S}, G, p)$
return $\mathrm{GNN}_\theta(\cdot)$

nomenon under the IC model and addressed the problem of maximizing the probability that the influence remains below a certain threshold, employing a greedy algorithm based on Monte Carlo simulation. Zheng and Pan (2018) employed a greedy algorithm to remove bridge end nodes to prevent propagation to other communities. Zhu et al. (2021a) removed node groups from given candidate node groups in an acyclic graph under a propagation model that extends the IC model by incorporating ECE (echo chamber effect). Xie et al. (2023) accelerated the influence estimation in the greedy algorithm using a dominator tree. Ni, Zhu, and Wang (2023) considered protecting specific nodes in multi-social networks by forcing them not to be activated.

Edge removal: The following works considered influence minimization with edge removal. Kimura, Saito, and Motoda (2009) minimized the average (or maximum) influence when each node is a seed set under the IC model using a greedy algorithm and accelerated the process with the bond percolation method. Tong et al. (2012) minimized the eigenvalue of the graph under the susceptible-infectious-susceptible (SIS) model. Khalil, Dilkina, and Song (2014) minimized the influence when one of the nodes in the given source set became a seed node under the LT model and employed a greedy algorithm that quickly calculates marginal loss using a live-edge tree. Yao et al. (2014) applied a greedy algorithm to the influence minimization problem identical to the one considered by us (i.e., Problem 1). Yan et al. (2019) addressed a problem identical to ours but with the condition of acyclic graphs. They analyzed the problem from the perspective of marginal decrement and proposed a heuristic algorithm based on propagation ability. Yi et al. (2022) accelerated the greedy algorithm in the G-SIR model by using reverse influence sampling. Zareie and Sakellariou (2022) minimized the spreading ability of the graph when the seed set is not given. Wang et al. (2020) proposed a robust sampling-based greedy algorithm to protect a given target node in the LT model. Jiang et al. (2022) protected given target nodes in the SIR model, by identifying critical edges through sampling paths from the seed nodes to the target nodes

Active defense: The following works proposed active defense to counter negative propagation. Budak, Agrawal, and El Abbadi (2011) used a Monte Carlo simulation-based greedy algorithm in the IC model and compared it with some heuristics. He et al. (2012) estimated the influence of each node by using a locally directed acyclic graph in the LT model. Fan et al. (2013) selected bridge ends to protect other communities from propagation using a greedy algorithm in the opportunistic one-activate-one model and deterministic one-activate-many model. Luo et al. (2014) employed a Monte Carlo simulation-based greedy algorithm in the continuous-time multiple campaign diffusion model. Zhu, Li, and Zhang (2016) efficiently calculated single-hop spread to estimate influence in the IC model. Tong and Du (2019) developed a hybrid sampling process in the IC model that attaches high weights to the users vulnerable to misinformation. Hosni, Li, and Ahmad (2019) used a greedy algorithm that simultaneously performs node blocking and active defense.

E Additional details of baseline methods

1. **MDS** (Yan et al. 2019) estimates the incremental change in the influence when each edge is removed, and greedily chooses b edges w.r.t. the incremental changes. In Alg. 4, we provide the pseudo-code of MDS. The edges are chosen according to the influenced probability and rumor-spread ability of their endpoints, with incremental updates.
2. **BPM** (Kimura, Saito, and Motoda 2009) uses the bond percolation method (BPM) to estimate the importance of edges, and removes the top-b edges w.r.t. importance scores. BPM does not consider specific seed nodes.
3. **Modified BPM (MBPM)** is a modified version of BPM that considers specific seed nodes. Specifically, we count the number of nodes reachable from the seed set S, in a sampled graph according to the activation probabilities p. We provide its pseudo-code in Alg. 5.
4. **RIS** (Yi et al. 2022) randomly samples a node and an activated graph each round, checks for each edge whether the node is reachable from the set of seeds without traversing the edge, and removes bottom-b edges w.r.t. the success rate. Let $\tilde{\sigma}(S; G, p)$ be the number of infected nodes (including recovered nodes for the G-SIR model (Yi et al. 2022)) except for the seed set. According to Proposition 8.1 in (Yi et al. 2022), a $(1 \pm \epsilon)$-approximation of $\tilde{\sigma}(S; G, p)$ is obtained with $O(\epsilon^{-2} n \log n)$ rounds of sampling an activated graph and a node. Therefore, as the error term ϵ decreases, a larger number of rounds is required. In our experiments, we conduct $0.1 \cdot \epsilon^{-2} n \log n$ rounds for $\epsilon = 0.2$, 0.4, and 0.6, respectively.

F Additional details of experimental settings

DIFFIM: We implemented DIFFIM in Python with the PyG library (Fey and Lenssen 2019) using the Adam op-

timizer (Kingma and Ba 2014). The initial node features were one-dimensional binary: 1 if the node is a seed node, and 0 otherwise. For GNN training (Alg. 3), the learning rate and its decaying rate were optimized by Optuna (Akiba et al. 2019) with 2000 epochs.

Hardware: All the versions of DIFFIM were run on a machine with 2.10GHz Intel® Xeon® Silver 4210R processors and RTX2080Ti GPUs. A single GPU was used for each experiment. The baseline methods did not use GPUs, and they were run on a machine with more powerful CPUs, 3.70GHz Intel® Core™ i9-10900KF processors.

G Additional experimental results

G.1 Additional results for Q1. Performances

We reported the effectiveness (the reduced ratio of influence) and the running time, along with the standard deviations, for each method, for each budget $b \in \{3, 5, 7, 10\}$, in Fig. 5 and Table 7. The standard deviations were computed over different seed sets, which could be high since the seed sets could be highly different.

G.2 Budget scalability

We conducted experiments with all the considered methods with budget b increasing from 1 to 10. In Table 8, for each method and each dataset, we reported the minimum value of b with which the method ran out of time (i.e., took more than one hour on a single seed set), where "." indicates that the method did not run out of time even with $b = 10$. The methods whose outputs do not depend on the seed set were not included. Among the versions of DIFFIM, only DIFFIM failed on the largest dataset WC. Among the baseline methods, MBPM had the lowest budget scalability, and GREEDY did not scale well on large graphs.

G.3 Variants of DIFFIM

For DIFFIM+, one can optimize $\tilde{r}$ and then choose the bottom-b edges with the lowest $\tilde{r}$ values together instead of removing edges one by one for each n_{ep} epochs. In Fig. 6, we reported the effectiveness of the variant (selecting edges together after bn_{ep} epochs) compared to original DIFFIM+ (selecting edges one by one) with different budgets $b \in [10]$ for each dataset. The effectiveness of the variant of DIFFIM+ was nearly the same as the original DIFFIM+.

For DIFFIM++, one can choose the top-b edges with the largest gradient together. In Fig. 7, we reported the effectiveness of the variant (selecting edges together instantly) compared to the original DIFFIM++ (selecting edges one by one). The effectiveness of the variant of DIFFIM++ was noticeably lower than that of original DIFFIM++, except on the CL dataset.

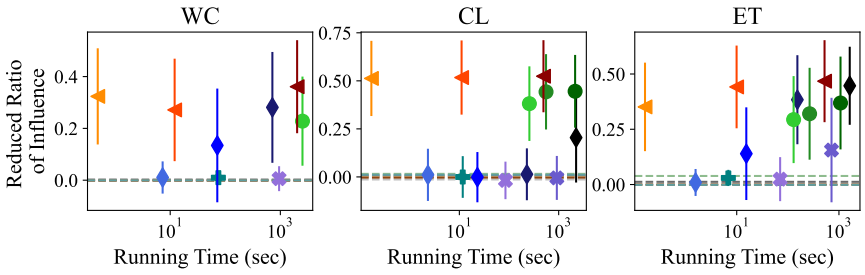
G.4 Estimation errors along training

As shown in Fig. 8, the estimation errors on the validation set decreased as the training proceeded. Overall, the trained GCNs achieved good estimation quality, with errors at most 4.5% of the ground-truth influence.

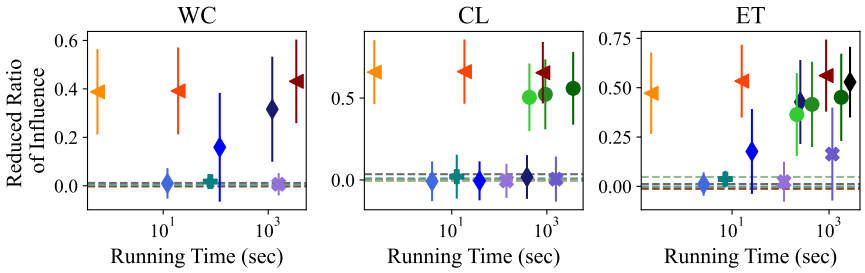
G.5 Performance on large-scale datasets

We additionally conducted experiments on three large-scale datasets: cit-HepTh (Leskovec, Kleinberg, and Faloutsos 2005; Gehrke, Ginsparg, and Kleinberg 2003), email-EuAll (Leskovec, Kleinberg, and Faloutsos 2007), and twitter (Leskovec and Mcauley 2012). All these datasets are available at SNAP (Leskovec and Krevl 2014). We provided their basic statistics in Table 9. Due to the absence of realistic activation probabilities, for these datasets, we used the weighted cascade model (Kempe, Kleinberg, and Tardos 2003), a special case of the IC model where the activation probability of each edge from node u to v is 1 divided by the in-degree of v.

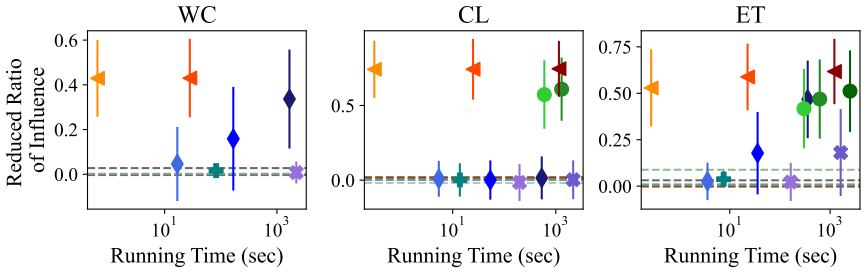
The results with a budget of $b = 5$ and $b = 10$ are presented in Figure 9. The proposed method consistently and significantly outperformed the baseline methods that completed within the time and memory limits. Here, BPM, DIFFIM (naive), RIS, and MBPM-100000 ran out of time.



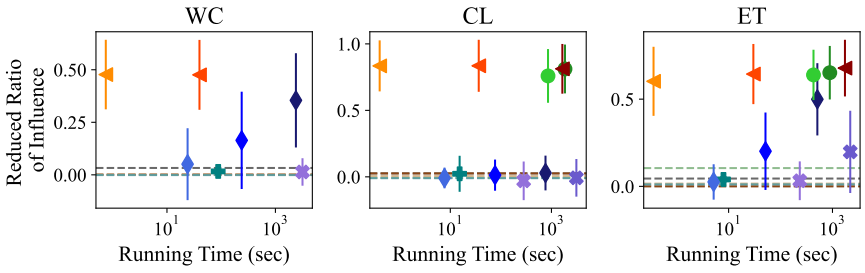
(a) $b = 3$



(b) $b = 5$



(c) $b = 7$



(d) $b = 10$

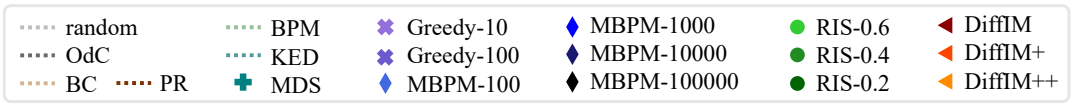


Figure 5: The effectiveness (the reduced ratio of influence) and running time of each method, with budget $b \in \{3, 5, 7, 10\}$. The error bars represent one standard deviation. The superiority of DIFFIM is valid with all the values of b.

Algorithm 4: MDS

Input: (1) $G = (V, E)$: an input graph
(2) $p : E \to [0, 1]$: activation probabilities
(3) $S \subseteq V$: a seed set
(4) b: an edge-removal budget
(5) h: the number of propagation hops
Output: $\mathcal{E} \subset E$: a set of edges chosen to be removed

$\texttt{prob}(v) \leftarrow \pi(v; G, p, S)$ ▷ MC simulation
$A \leftarrow$ the adjacent matrix of G
$\texttt{rsa} \leftarrow \sum_{0 \le i \le h} A^i \mathbf{1}$
for $i = 1, 2, \ldots, b$ **do**
  $s_e \leftarrow (1 - \texttt{prob}(u)) \cdot (\texttt{rsa}(v) + \texttt{rsa}(u)), \forall (v, u) \in E$
  $e \leftarrow \arg\max_{e \in E} s_e$
  $\texttt{rsa} \leftarrow \texttt{update\_rsa}(G, p, e, \texttt{rsa})$
  $\texttt{prob} \leftarrow \texttt{update\_prob}(G, p, e, \texttt{prob})$
  $E \leftarrow E \setminus \{e\}; \mathcal{E} \leftarrow \mathcal{E} \cup \{e\}$
return $\mathcal{E}$
Function *update_prob* $(G, p, e = (s, t), \texttt{prob})$**:**
  $\Delta\texttt{prob}(v) \leftarrow 0, \forall v \in V$
  $\Delta\texttt{prob}(t) \leftarrow \frac{p(s,t) \cdot \texttt{prob}(s) \cdot (1 - \texttt{prob}(t))}{1 - p(s,t) \cdot \texttt{prob}(s)}$
  $U \leftarrow \{t\}; V \leftarrow \{t\}$
  while $U \neq \emptyset$ **do**
    $U' \leftarrow \emptyset$
    foreach $(u, v) \in E$ *such that* $u \in U, v \in V$ **do**
      $\Delta\texttt{prob}(v) \leftarrow \Delta\texttt{prob}(v) + \frac{p(u,v) \cdot \Delta\texttt{prob}(u) \cdot (1 - \texttt{prob}(v))}{1 - p(u,v) \cdot \texttt{prob}(u)}$
      if $v \notin V$ **then**
        $U' \leftarrow U' \cup \{v\}; V \leftarrow V \cup \{v\}$
    $U \leftarrow U'$
  $\texttt{prob}(v) \leftarrow \texttt{prob}(v) - \Delta\texttt{prob}(v), \forall v \in V$
  return $\texttt{prob}$
Function *update_rsa* $(G, p, e = (s, t), \texttt{rsa})$**:**
  $\Delta\texttt{rsa}(t) \leftarrow p(s, t) \cdot \texttt{rsa}(t)$
  $U \leftarrow \{s\}; V \leftarrow \{s\}$
  while $U \neq \emptyset$ **do**
    $U' \leftarrow \emptyset$
    foreach $(v, u) \in E$ *such that* $v \in V, u \in U$ **do**
      $\Delta\texttt{rsa}(v) \leftarrow \Delta\texttt{rsa}(v) + p(v, u) \cdot \Delta\texttt{rsa}(u)$
      if $v \notin V$ **then**
        $U' \leftarrow U' \cup \{v\}; V \leftarrow V \cup \{v\}$
    $U \leftarrow U'$
  $\texttt{rsa}(v) \leftarrow \texttt{rsa}(v) - \Delta\texttt{rsa}(v), \forall v \in V$
  return $\texttt{rsa}$

Algorithm 5: Modified BPM

Input: (1) $G = (V, E)$: an input graph
(2) $p : E \to [0, 1]$: activation probabilities
(3) $S \subseteq V$: a seed set
(4) b: an edge-removal budget
(5) d: the number of samplings
Output: $\mathcal{E} \subset E$: a set of edges chosen to be removed

$\mathcal{E} \leftarrow \emptyset$
for $i = 1, 2, \ldots, b$ **do**
  $s_e \leftarrow 0, n_e \leftarrow 0, \forall e \in E$
  for $j = 1, 2, \ldots, d$ **do**
    $l_e \sim \text{Bernoulli}(p(e)), \forall e \in E$
    $E' \leftarrow \{e \in E \colon l_e = 1\}$
    $G' \leftarrow (V, E')$
    $n_S \leftarrow$ the number of nodes reachable from S on G'
    foreach $e \in E$ *such that* $l_e = 0$ **do**
      $s_e \leftarrow s_e + n_S; n_e \leftarrow n_e + 1$
  foreach $e \in E$ **do**
    if $n_e \geq 1$ **then**
      $s_e \leftarrow s_e / n_e$
    else
      $s_e \leftarrow \infty$
  $e \leftarrow \arg\min_e s_e$
  $E \leftarrow E \setminus \{e\}; \mathcal{E} \leftarrow \mathcal{E} \cup \{e\}$
return $\mathcal{E}$

method	$b = 3$			$b = 5$		
	WC	CL	ET	WC	CL	ET
RANDOM	0.0016 (0.0214)	-0.0133 (0.1105)	0.0010 (0.0182)	-0.0045 (0.0216)	0.0010 (0.1188)	0.0081 (0.0158)
OdC	0.0022 (0.0225)	0.0063 (0.1077)	0.0123 (0.0207)	0.0117 (0.0238)	0.0352 (0.1534)	0.0119 (0.0407)
BC	-0.0006 (0.0226)	0.0006 (0.0860)	-0.0011 (0.0208)	0.0019 (0.0220)	-0.0057 (0.0992)	-0.0028 (0.0206)
PR	-0.0017 (0.0175)	-0.0036 (0.1204)	-0.0002 (0.0187)	-0.0017 (0.0201)	-0.0010 (0.1077)	-0.0124 (0.0182)
BPM	**O.O.T**	0.0159 (0.1221)	0.0387 (0.0471)	**O.O.T**	-0.0021 (0.1074)	0.0477 (0.0527)
KED	-0.0006 (0.0174)	0.0109 (0.1162)	-0.0015 (0.0180)	0.0029 (0.0178)	0.0097 (0.1317)	-0.0029 (0.0556)
MDS	0.0098 (0.0254)	-0.0004 (0.1085)	0.0298 (0.0366)	0.0173 (0.0229)	0.0202 (0.1341)	0.0368 (0.0421)
MBPM-100	0.0105 (0.0622)	0.0108 (0.1358)	0.0093 (0.0610)	0.0098 (0.0630)	-0.0076 (0.1215)	0.0116 (0.0594)
MBPM-1000	0.1342 (0.2195)	-0.0013 (0.1308)	0.1396 (0.2096)	0.1589 (0.2245)	-0.0052 (0.1187)	0.1767 (0.2150)
MBPM-10000	0.2811 (0.2140)	0.0136 (0.1350)	0.3836 (0.2014)	0.3160 (0.2170)	0.0183 (0.1335)	0.4273 (0.2127)
MBPM-100000	**O.O.T**	0.2046 (0.2338)	0.4468 (0.1766)	**O.O.T**	**O.O.T**	0.5284 (0.1790)
GREEDY-10	0.0059 (0.0481)	-0.0182 (0.0976)	0.0244 (0.0995)	0.0062 (0.0459)	-0.0052 (0.1040)	0.0236 (0.1011)
GREEDY-100	**O.O.T**	-0.0052 (0.1144)	0.1560 (0.2363)	**O.O.T**	0.0052 (0.1374)	0.1635 (0.2356)
RIS-0.6	0.2280 (0.1720)	0.3813 (0.1941)	0.2937 (0.1968)	**O.O.T**	0.5045 (0.2064)	0.3641 (0.2102)
RIS-0.4	**O.O.T**	0.4428 (0.1960)	0.3202 (0.2078)	**O.O.T**	0.5227 (0.2136)	0.4156 (0.2163)
RIS-0.2	**O.O.T**	0.4456 (0.1893)	0.3688 (0.2100)	**O.O.T**	0.5591 (0.2217)	0.4513 (0.2212)
DIFFIM	0.3609 (0.1796)	0.5240 (0.1876)	0.4675 (0.1855)	0.4311 (0.1726)	0.6547 (0.1877)	0.5613 (0.1828)
DIFFIM+	0.2713 (0.1978)	0.5172 (0.1930)	0.4415 (0.1871)	0.3914 (0.1796)	0.6614 (0.1957)	0.5332 (0.1845)
DIFFIM++	0.3236 (0.1856)	0.5125 (0.1956)	0.3512 (0.2002)	0.3876 (0.1759)	0.6583 (0.1948)	0.4718 (0.2066)

method	$b = 7$			$b = 10$		
	WC	CL	ET	WC	CL	ET
RANDOM	-0.0011 (0.0187)	-0.0187 (0.1244)	0.0014 (0.0210)	0.0002 (0.0201)	-0.0072 (0.1150)	0.0005 (0.0182)
OdC	0.0275 (0.0330)	0.0106 (0.1120)	0.0315 (0.0439)	0.0327 (0.0338)	0.0263 (0.1294)	0.0448 (0.0494)
BC	0.0027 (0.0194)	0.0196 (0.1118)	0.0090 (0.0557)	0.0020 (0.0234)	0.0054 (0.1013)	0.0158 (0.0534)
PR	-0.0035 (0.0225)	0.0187 (0.1071)	-0.0020 (0.0203)	0.0006 (0.0239)	0.0257 (0.1377)	0.0002 (0.0218)
BPM	**O.O.T**	0.0016 (0.0980)	0.0882 (0.1145)	**O.O.T**	-0.0085 (0.1030)	0.1047 (0.1432)
KED	-0.0010 (0.0158)	0.0017 (0.1316)	0.0079 (0.0569)	-0.0012 (0.0197)	-0.0100 (0.1319)	0.0113 (0.0579)
MDS	0.0160 (0.0244)	0.0015 (0.1112)	0.0372 (0.0442)	0.0163 (0.0258)	0.0230 (0.1354)	0.0396 (0.0424)
MBPM-100	0.0461 (0.1659)	0.0091 (0.1202)	0.0258 (0.1008)	0.0507 (0.1711)	-0.0080 (0.0778)	0.0252 (0.1015)
MBPM-1000	0.1587 (0.2318)	0.0009 (0.1315)	0.1778 (0.2219)	0.1641 (2315)	0.0128 (0.1172)	0.2014 (0.2220)
MBPM-10000	0.3365 (0.2215)	0.0153 (0.1437)	0.4673 (0.2089)	0.3545 (2243)	0.0289 (0.1307)	0.4991 (0.2072)
MBPM-100000	**O.O.T**	**O.O.T**	**O.O.T**	**O.O.T**	**O.O.T**	**O.O.T**
GREEDY-10	0.0076 (0.0493)	-0.0152 (0.1241)	0.0224 (0.1024)	0.0136 (0.0654)	-0.0300 (0.1453)	0.0321 (0.1111)
GREEDY-100	**O.O.T**	0.0030 (0.1299)	0.1809 (0.2340)	**O.O.T**	-0.0074 (0.1414)	0.1977 (0.2358)
RIS-0.6	**O.O.T**	0.5732 (0.2292)	0.4172 (0.2134)	**O.O.T**	0.7590 (0.2020)	0.6390 (0.1443)
RIS-0.4	**O.O.T**	0.6093 (0.2117)	0.4688 (0.2131)	**O.O.T**	0.8113 (0.1836)	0.6517 (0.1534)
RIS-0.2	**O.O.T**	**O.O.T**	0.5116 (0.2195)	**O.O.T**	**O.O.T**	**O.O.T**
DIFFIM	**O.O.T**	0.7455 (0.1871)	0.6171 (0.1757)	**O.O.T**	0.8124 (0.1870)	0.6780 (0.1625)
DIFFIM+	0.4301 (0.1754)	0.7423 (0.2034)	0.5881 (0.1794)	0.4758 (0.1663)	0.8352 (0.1957)	0.6439 (0.1725)
DIFFIM++	0.4289 (0.1714)	0.7417 (0.1914)	0.5286 (0.2078)	0.4772 (0.1655)	0.8346 (0.1919)	0.6023 (0.1980)

Table 7: The effectiveness (the reduced ratio of influence) of each method with the standard deviations, with budget $b \in \{3, 5, 7, 10\}$. O.O.T denotes out-of-time, i.e., the method does not terminate within one hour on a single seed set in the corresponding setting.

methods	WC	CL	ET
MDS	.	.	.
MBPM-10000	.	.	.
MBPM-100000	3	5	7
GREEDY-10	.	.	.
GREEDY-100	2	.	.
RIS-0.6	5	.	.
RIS-0.4	2	.	.
RIS-0.2	1	6	10
DIFFIM	6	.	.
DIFFIM+	.	.	.
DIFFIM++	.	.	.

Table 8: The minimum budget $b \leq 10$ with which each method runs out of time. Each cell with "." implies that the method did not run out of time even with $b = 10$.

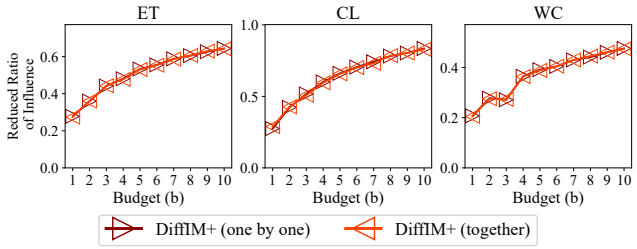


Figure 6: The effectiveness of the variant (selecting edges together after bn_{ep} epochs) of DIFFIM+ compared to original DIFFIM+ (selecting edges one by one) with respect to budgets $b \in [10]$ for each dataset. The effectiveness of the variant was nearly the same as the original DIFFIM+.

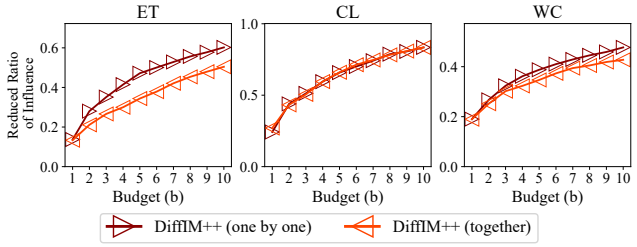


Figure 7: The effectiveness of the variant (selecting edges together) of DIFFIM++ compared to original DIFFIM++ (selecting edges one by one) with respect to budgets $b \in [10]$ for each dataset. The effectiveness of the variant is lower than that of the original DIFFIM+, except on the CL dataset.

dataset	abbr.	$\lvert V\rvert$	$\lvert E\rvert$
cit-HepTh	HT	27,770	352,807
email-EuAll	EA	265,214	420,045
twitter	TW	81,306	1,768,149

Table 9: The basic statistics of the large-scale datasets.

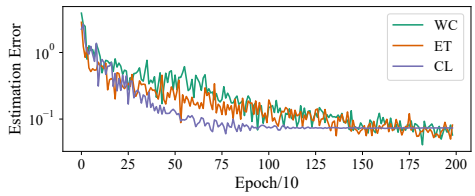


Figure 8: The average validation estimation errors in estimating influence decreased as GCN training proceeds.

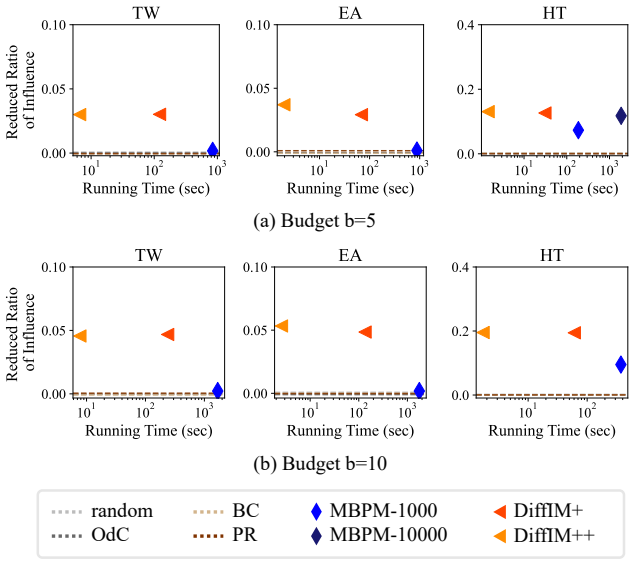


Figure 9: The effectiveness (the reduced ratio of influence) and running time of each method, with budget $b = 5$ (top) and $b = 10$ (bottom) on the large-scale datasets

References

Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; and Koyama, M. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *KDD*.

Budak, C.; Agrawal, D.; and El Abbadi, A. 2011. Limiting the spread of misinformation in social networks. In *WWW*.

Chang, C.-W.; Yeh, M.-Y.; and Chuang, K.-T. 2016. On the guarantee of containment probability in influence minimization. In *ASONAM*.

Fan, L.; Lu, Z.; Wu, W.; Thuraisingham, B.; Ma, H.; and Bi, Y. 2013. Least cost rumor blocking in social networks. In *ICDCS*.

Fey, M.; and Lenssen, J. E. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv:1903.02428*.

Gehrke, J.; Ginsparg, P.; and Kleinberg, J. 2003. Overview of the 2003 KDD Cup. *Acm Sigkdd Explorations Newsletter*, 5(2): 149–151.

He, X.; Song, G.; Chen, W.; and Jiang, Q. 2012. Influence blocking maximization in social networks under the competitive linear threshold model. In *SDM*.

Hosni, A. I. E.; Li, K.; and Ahmad, S. 2019. DARIM: Dynamic approach for rumor influence minimization in online social networks. In *NeurIPS*.

Jiang, Z.; Chen, X.; Ma, J.; and Philip, S. Y. 2022. RumorDecay: rumor dissemination interruption for target recipients in social networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(10): 6383–6395.

Kempe, D.; Kleinberg, J.; and Tardos, É. 2003. Maximizing the spread of influence through a social network. In *KDD*.

Khalil, E. B.; Dilkina, B.; and Song, L. 2014. Scalable diffusion-aware optimization of network topology. In *KDD*.

Kimura, M.; Saito, K.; and Motoda, H. 2009. Blocking links to minimize contamination spread in a social network. *ACM Transactions on Knowledge Discovery from Data*, 3(2): 1–23.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980*.

Leskovec, J.; Kleinberg, J.; and Faloutsos, C. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD*.

Leskovec, J.; Kleinberg, J.; and Faloutsos, C. 2007. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1): 2–es.

Leskovec, J.; and Krevl, A. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

Leskovec, J.; and Mcauley, J. 2012. Learning to discover social circles in ego networks. In *NeurIPS*.

Luo, C.; Cui, K.; Zheng, X.; and Zeng, D. 2014. Time critical disinformation influence minimization in online social networks. In *JISIC*.

Ni, P.; Zhu, J.; and Wang, G. 2023. Misinformation influence minimization by entity protection on multi-social networks. *Applied Intelligence*, 53(6): 6401–6420.

Tong, G. A.; and Du, D.-Z. 2019. Beyond uniform reverse sampling: A hybrid sampling technique for misinformation prevention. In *INFOCOM*.

Tong, H.; Prakash, B. A.; Eliassi-Rad, T.; Faloutsos, M.; and Faloutsos, C. 2012. Gelling, and melting, large graphs by edge manipulation. In *CIKM*.

Vinterbo, S. A. 2002. A note on the hardness of the k-ambiguity problem. *Technical Report*.

Wang, S.; Zhao, X.; Chen, Y.; Li, Z.; Zhang, K.; and Xia, J. 2013. Negative influence minimizing by blocking nodes in social networks. In *AAAI Workshops*.

Wang, X.; Deng, K.; Li, J.; Yu, J. X.; Jensen, C. S.; and Yang, X. 2020. Efficient targeted influence minimization in big social networks. *World Wide Web*, 23(4): 2323–2340.

Xie, J.; Zhang, F.; Wang, K.; Lin, X.; and Zhang, W. 2023. Minimizing the Influence of Misinformation via Vertex Blocking. *arXiv:2302.13529*.

Yan, R.; Li, Y.; Wu, W.; Li, D.; and Wang, Y. 2019. Rumor blocking through online link deletion on social networks. *ACM Transactions on Knowledge Discovery from Data*, 13(2): 1–26.

Yao, Q.; Zhou, C.; Xiang, L.; Cao, Y.; and Guo, L. 2014. Minimizing the negative influence by blocking links in social networks. In *ISCTCS*.

Yi, Y.; Shan, L.; Paré, P. E.; and Johansson, K. H. 2022. Edge deletion algorithms for minimizing spread in sir epidemic models. *SIAM Journal on Control and Optimization*, 60(2): S246–S273.

Zareie, A.; and Sakellariou, R. 2022. Rumour spread minimization in social networks: A source-ignorant approach. *Online Social Networks and Media*, 29: 100206.

Zheng, J.; and Pan, L. 2018. Least cost rumor community blocking optimization in social networks. In *SSIC*.

Zhu, J.; Ni, P.; Wang, G.; and Li, Y. 2021. Misinformation influence minimization problem based on group disbanded in social networks. *Information Sciences*, 572: 1–15.

Zhu, Y.; Li, D.; and Zhang, Z. 2016. Minimum cost seed set for competitive social influence. In *INFOCOM*.