Data Mining and Knowledge Discovery (2025) 39:55
https://doi.org/10.1007/s10618-025-01137-z

Inductive influence estimation and maximization over unseen social networks under two diffusion models

Jihoon Ko[1] · Sojeong Kim[1] · Kyuhan Lee[1] · Shinhwan Kang[1] · Dongyeong Hwang[1] · Kijung Shin[1] · Noseong Park[2]

Received: 24 March 2024 / Accepted: 12 July 2025

Abstract
Influence estimation (IE) and influence maximization (IM) are among the most extensively studied problems in social network analysis. Assuming diffusion (i.e., the spread of diseases) within a social network, IE aims to estimate the influence (i.e., the number of infected nodes) for a given set of seeds; and IM aims to identify a given number of seed nodes that maximize the influence. For both IE and IM, widely-adopted strategies involve repeating Monte Carlo (MC) simulations of diffusion over and over for various seed sets, which is computationally expensive. In this work, we present Monte Carlo Simulator+ (MONSTOR+), an inductive machine learning method designed to estimate the influence of given seed-node sets in social networks under two diffusion models—the independent cascade (IC) model and the linear threshold (LT) model. Due to its inductive nature, MONSTOR+ is applicable to seed-node sets and social networks not included in the training data. MONSTOR+, with its ability to accurately estimate influence through a single forward pass, can greatly accelerate existing IM algorithms by replacing repeated MC simulations. In our experiments, MONSTOR+ exhibits high IE accuracy, achieving 0.955 or higher Pearson and Spearman correlation coefficients in unseen real-world social networks. Notably, MONSTOR+ is about 5 to 3000 times faster than repeated MC simulations with similar IE accuracy. For IM problems, IM algorithms equipped with MONSTOR+ are more accurate than state-of-the-art competitors in 81.5 and 77.8% of IM use cases under the IC model and LT model, respectively.

Keywords Social network analysis · Influence estimation · Influence maximization · Graph neural networks

Jihoon Ko and Sojeong Kim contributed equally to this work.

Extended author information available on the last page of the article

Published online: 28 July 2025

1 Introduction

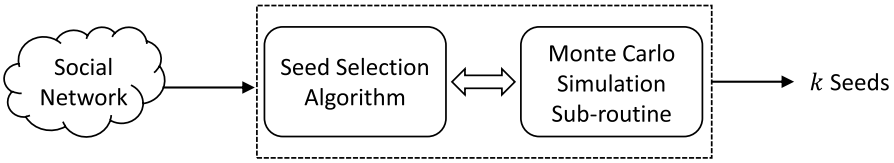
Diffusion within social networks, including the spread of new technologies, diseases, and news, is pervasive. In response, the *influence estimation* (IE) and *influence maximization* (IM) problems have received considerable attention, with applications ranging from viral marketing (Chen et al. 2010; Domingos and Richardson 2001), targeted advertisement (Li et al. 2015), and socio-political campaigns (Sankar and Kumar 2016). Assuming diffusion (i.e., the spread of diseases) within a social network, IE aims to accurately estimate the influence (i.e., the number of infected nodes) for given seed nodes; and IM aims to find a certain number of seed nodes that maximize influence.

For IE, repeated Monte Carlo (MC) simulations of diffusion from the given seed set are widely used; and many IM algorithms (e.g., Greedy (Kempe et al. 2003), CELF (Goyal et al. 2011a), and UBLF (Zhou et al. 2013)) require repeated MC simulations for a large number of seed sets. An MC simulation takes $O(|\mathcal{E}|)$ time, where $|\mathcal{E}|$ is the number of edges, and estimating the influence of a seed set via d simulations takes $O(d|\mathcal{E}|)$ time, which is one of the main performance bottlenecks of IM algorithms. In (Kempe et al. 2003; Zhou et al. 2013), d is set to 10, 000.

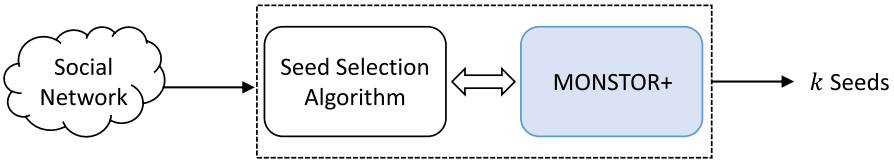
In this work, we propose Monte Carlo Simulator+ (MONSTOR+), an inductive machine learning approach for estimating the influence of given seed nodes.[1] MONSTOR+ offers several advantages. First, MONSTOR+ operates effectively under both the independent cascade (IC) and linear threshold (LT) models, which are among the most widely-used diffusion models. Second, by leveraging an inductive machine learning model based on graph neural networks (GNNs) and features, MONSTOR+ is capable of producing inference for seed-node sets and social networks not encountered during training. Third, compared to its preliminary version named MONSTOR, MONSTOR+ incorporates auxiliary structural node features and an advanced pooling function, resulting in improved estimation accuracy. Lastly, MONSTOR+ significantly speeds up existing IM methods by replacing repeated MC simulations (Fig. 1).

For evaluation, we conduct experiments with three real-world social networks. One strong point in our experiments is that we use real activation probabilities of edges, which are calculated from retweet logs. That is, we weight each directed edge (u, v) with the real probability that user u influences v. Note that most previous studies on influence maximization simply used random, uniform, and degree-based prob-

[1] This paper is an extended version of (Ko et al. 2020), where we introduced a preliminary version of MONSTOR+. The preliminary version, MONSTOR, is a pioneering inductive machine learning approach for estimating the influence of given seed sets under the IC model. In this extension, we present MONSTOR+, which enhances MONSTOR by incorporating auxiliary structural node features and an advanced pooling function (Sect. 3.3). Both enhancements lead to improvements in accuracy (Sect. 4.4). Furthermore, while MONSTOR is tailored for the IC model, MONSTOR+ is not restricted to any specific diffusion model. We enhance the comprehensiveness of our experiments by including an additional diffusion model (the LT model), four more competitors (Tang et al. 2014, 2015; Goyal et al. 2011b; Panagopoulos et al. 2023), an ablation study (Sect. 4.4), and numerical comparisons with MC simulations (Sect. 4.2.2). Although our main experiments focus on the IC and LT models, we further apply MONSTOR+ to the G-SIR model (Yi et al. 2022) (Appendix A.3). Additionally, to examine its scalability, we evaluate MONSTOR+ on a large-scale dataset (Appendix A.4).



(a) Influence maximization based on repeated MC simulation



(b) Influence maximization based on MONSTOR+

Fig. 1 Comparison of influence maximization approaches: **a** MC simulation-based approaches, and **b** our approach equipped with the proposed Monte Carlo Simulator+ (MONSTOR+)

abilities (Nguyen et al. 2016; Jung et al. 2012; Wang et al. 2012; Goyal et al. 2011a; Zhou et al. 2013), which are different from real ones.

In our experiments, MONSTOR+ yielded accurate estimations of influence, exhibiting near-perfect correlation with ground-truth values. Additionally, it achieves significant time savings, performing 5 to 3000 times faster than repeated Monte Carlo simulations for similar IE accuracy. Regarding IM, simulation-based IM algorithms (Kempe et al. 2003; Goyal et al. 2011a; Zhou et al. 2013) equipped with MONSTOR+ yielded influence maximization results nearly on par with those of the original algorithms based on MC simulations, under both diffusion models. Moreover, it achieves higher influences than state-of-the-art non-simulation-based IM algorithms (Tang et al. 2014, 2015; Nguyen et al. 2016; Jung et al. 2012; Wang et al. 2012) in 22 out of 27 cases under the IC model and in 7 out of 9 cases under the LT model.

In summary, MONSTOR+ exhibits the following strengths:

- *Inductive*: It is designed to be applicable to seed-node sets and social networks not encountered during training.
- *Fast and accurate*: Its estimates exhibit near-perfect correlation with ground-truth influences. It achieves a speedup of 5 to 3000 times, compared to MC simulations, while retaining similar estimation accuracy.
- *Applicable to influence maximization*: Integrated into IM methods, MONSTOR+ performed best in 81.5% and 77.8% of the cases considered under the IC model and the LT model, respectively, among the 10 compared methods.

For **reproducibility**, we make the source code and datasets used in this paper available at https://github.com/SojeongKim00/MONSTOR-plus.

The rest of this paper is organized as follows. In Sect. 2, we introduce some concepts and notions related to our problem. In Sect. 3, we describe the overall workflow and the detailed design of MONSTOR+. In Sect. 4, we present experimental results. After reviewing related works in Sect. 5, we present conclusions in Sect. 6.

2 Concepts and problem definition

In this section, we introduce two diffusion models with related concepts, and we define two problems that are considered in this work. In both models, we assume a social network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the node set $\mathcal{V} = \{1, \cdots, |\mathcal{V}|\}$ and the edge set $\mathcal{E}$, and its *adjacency matrix* $\mathrm{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ where $A_{uv} = 1$ if and only if $(u, v) \in E$.

2.1 Independent cascade (IC) model

Model definition. Under the *independent cascade* (IC) model (Kempe et al. 2003), each infected node *u* attempts **once** to activate (i.e., directly infect) each neighbor *v*, and the probability of success is $p_{(u,v)}$, as defined below.

Definition 1 (Activation Probability) The *activation probability* $p_{(u,v)}$ from *u* to *v* is the success probability that the node *u* activates its neighbor *v* when *u* is infected.

The adjacency matrix when weighting each directional edge (*u*, *v*) by $p_{(u,v)}$ is called the *activation probability matrix* $\mathrm{P} \in [0,1]^{|\mathcal{V}|\times|\mathcal{V}|}$. That is, each (*u*, *v*)-th entry of P is $p_{(u,v)}$.

Given an activation probability matrix and a set of *seed nodes* (i.e., initially infected nodes), the IC model simulates the above activation process for each newly infected node until there are no newly infected nodes, and we define the *infection probability* of each node as follows:

Definition 2 (Infection Probability and Influence under the IC Model) Given an activation probability matrix $\mathrm{P} \in [0,1]^{|\mathcal{V}|\times|\mathcal{V}|}$ of a social network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a seed set $S \subseteq \mathcal{V}$, the *infection probability* $\rho(x; \mathrm{P}, S)$ for each node $x \in \mathcal{V}$ is defined as the probability that *x* is infected under the IC model when the seed set is *S*. The *influence* $\phi(S; \mathrm{P})$ of the seed set *S* is defined as the expected number of infected nodes, equivalent to the sum of the infection probabilities, i.e., $\phi(S; \mathrm{P}) = \sum_{x\in\mathcal{V}} \rho(x; \mathrm{P}, S)$.

Throughout the paper, we omit P and *S* in $\rho(x; \mathrm{P}, S)$ and use $\rho(x)$ when there is no ambiguity. Similarly, we use $\phi(S)$ when there is no ambiguity.

Inference of activation probabilities. As described above, the IC model requires activation probabilities as inputs, which are not directly observable. In this work, we infer activation probabilities from interaction logs (such as retweets) among users in a social network as follows:

1. Bernoulli Trial (BT):

$$p_{(u,v)} = \frac{|\text{actions}(u,*) \cap \text{actions}(v,*)|}{|\text{actions}(u,*)|},$$

2. Jaccard Index (JI):

$$p_{(u,v)} = \frac{|\text{actions}(u,*) \cap \text{actions}(*,v)|}{|\text{actions}(u,*) \cup \text{actions}(*,v)|},$$

3. Linear Probability (LP):

$$p_{(u,v)} = \frac{|\text{actions}(u,*) \cap \text{actions}(*,v)|}{|\text{actions}(*,v)|},$$

where $\text{actions}(x,*)$ denotes the set of actions (e.g., retweets and replies) done by node *x*, and $\text{actions}(*,x)$ denotes the set of actions whose object (e.g., author of retweeted tweets and recipient of replies) is *x*. We consider all these probabilities, and thus we define three different activation probability matrices, P_{BT}, P_{JI}, and P_{LP}, from a social network.

2.2 Linear threshold (LT) model

Model definition

Under the *linear threshold* (LT) model (Kempe et al. 2003), each node *v* becomes infected when the ratio of its infected neighbors exceeds its threshold. The threshold of each node is an independent random variable uniformly distributed between 0 and 1.

Given a social network and a set of seed nodes (i.e., initially infected nodes), the LT model proceeds until there are no newly infected nodes, and we define the *infection probability* of each node as follows:

Definition 3 (Infection Probability and Influence under the LT Model)

Given the adjacency matrix $\mathrm{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ of a social network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a seed set $S \subseteq \mathcal{V}$, the *infection probability* $\rho(x; \mathrm{A}, S)$ for each node $x \in \mathcal{V}$ is the probability that *x* is infected under the LT model when the seed set is *S*. As in the IC model, the *influence* $\phi(S; \mathrm{A})$ of the seed set *S* is defined as the expected number of infected nodes, equivalent to the sum of the infection probabilities, i.e., $\phi(S; \mathrm{A}) = \sum_{x\in\mathcal{V}} \rho(x; \mathrm{A}, S)$.
Throughout the paper, we omit A and *S* in $\rho(x; \mathrm{A}, S)$ and use $\rho(x)$ when there is no ambiguity. Similarly, we use $\phi(S)$ when there is no ambiguity.

2.3 Problem definition

Here, we offer formal definitions of the two problems addressed in this work: influence estimation (IE) and influence maximization (IM).

Problem 1 *(Influence Estimation (IE))*

- *Given*: (1) an activation probability matrix P (under the *IC model*) or an adjacent matrix A (under the *LT model*), and (2) a seed set S,
- *Estimate*: (1) the infection probability $\rho(x)$ of each node $x \in \mathcal{V}$, and (2) the *influence* $\phi(S)$ of S.

Problem 2 *(Influence Maximization (IM) (Kempe et al. 2003))*

- *Given*: (1) an activation probability matrix P (under the *IC model*) or an adjacent matrix A (under the *LT model*), and (2) the target number k of seed nodes (i.e., $k = |S|$),
- *Find* the set S of k seed nodes
- to *Maximize* the influence $\phi(S)$ (i.e., the expected number of infected nodes).

3 Proposed framework

In this section, we propose MONSTOR+, an inductive machine learning model for estimating the infection probability $\rho(x)$ of every node $x \in \mathcal{V}$ in a social network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that is not necessarily included in training data. Note that we can answer the IE problem by summing up the estimates and answer the IM problem by replacing MC simulations in simulation-based algorithms (e.g., (Kempe et al. 2003; Goyal et al. 2011a; Zhou et al. 2013)) with MONSTOR+.

Below, we introduce MONSTOR, a simplified version of MONSTOR+, and later extend it to MONSTOR+. For both methods, we begin by defining key concepts and outlining the overall workflow.

3.1 Key concepts and overall workflow

In this subsection, we outline the overall workflow in MONSTOR and MONSTOR+. To do so, we introduce key concepts used in both approaches.

Key concepts

We define a *step* of a diffusion model (the **IC** or **LT** model) as a one-hop cascade from newly infected nodes. Then, by definition, seed nodes are infected in the 0-th step, and those infected directly by seed nodes are infected in the 1-st step. Then, we let $\rho_i(x)$ denote the probability of node x being infected during the first i steps; and we call $\boldsymbol{\pi}_i := [\rho_i(1), \cdots, \rho_i(|\mathcal{V}|)]$ the *infection probability vector within each i-th step*. With these definitions, Proposition 1 follows.

Proposition 1 *The infection probability monotonically increases w.r.t. the step number i. That is, $\boldsymbol{\pi}_i \leq \boldsymbol{\pi}_{i+1}$ for all $i \geq 0$, or equivalently $\rho_i(x) \leq \rho_{i+1}(x)$ for all $i \geq 0$ and $x \in \mathcal{V}$.*

Similarly, we define $\boldsymbol{\pi} = [\rho(1), \cdots, \rho(|\mathcal{V}|)]$ for the infection probabilities at the end of the diffusion model. Note that $\rho_i(x) \approx \rho(x)$ and $\boldsymbol{\pi}_i \approx \boldsymbol{\pi}$ if i is sufficiently large; and $\rho_i(x) = \rho(x)$ and $\boldsymbol{\pi}_i = \boldsymbol{\pi}$ if i is greater than or equal to the longest path length in the input network.

Overall workflow

The overall workflow in MONSTOR and MONSTOR+ is as follows:

1. We collect one or more social networks $\{\mathcal{G}_1, \mathcal{G}_2, \cdots\}$.
2. From each $\mathcal{G}_j$, we collect the tuples $\{(\boldsymbol{\pi}_i, \boldsymbol{\pi}_{i-1}, \cdots, \boldsymbol{\pi}_{i-e}, \mathrm{P}_j) : i \geq e\}$ (in the case of the IC model) or $\{(\boldsymbol{\pi}_i, \boldsymbol{\pi}_{i-1}, \cdots, \boldsymbol{\pi}_{i-e}, \mathrm{A}_j) : i \geq e\}$ (in the case of the LT model), where $e > 1$ is a hyperparameter, after choosing a seed set $S \neq \emptyset$ randomly. P_j can be in BT, JI, or LP. We repeat this multiple times with different seed sets, as shown in Fig. 2. For detailed collection methods used in the experiments and details specific to our experimental setup, refer to Sect. 4.1.
3. We train a graph neural network (GNN)-based model M with the training data. It has l GNN layers, and it estimates $\boldsymbol{\pi}_i$ given $\boldsymbol{\pi}_{i-1}, \cdots, \boldsymbol{\pi}_{i-e}$. That is, M is trained to estimate a single step of the considered diffusion model (IC or LT). We present in detail two alternative implementations of M in the following subsections.
4. In the testing phase, we stack the pre-trained model M s times in a feed-forward manner to estimate $\boldsymbol{\pi}_s$ from $\boldsymbol{\pi}_0$. The output of the stacked model (i.e., the estimated $\boldsymbol{\pi}_s$) serves as the final estimate of the infection probabilities $\boldsymbol{\pi}$. For accurate estimation, the number of stacks s is carefully selected based on validation performance.[2] Hereinafter, MONSTOR(+) means the stacked GNN-based model, which is illustrated in Fig. 3, unless otherwise stated. In essence,

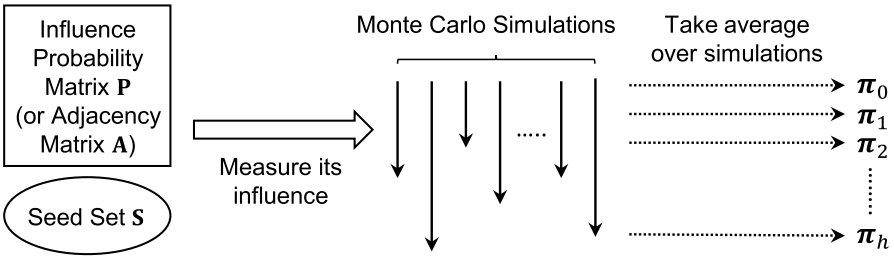


Fig. 2 How to build training data. Given a social network $\mathcal{G}$ (whose activation probability matrix is P under the IC model and adjacency matrix is A under the LT model) and a seed set, we perform multiple simulations and collect infection probability vectors $\boldsymbol{\pi}_0, \cdots, \boldsymbol{\pi}_h$, here h is either the convergence step (i.e., $\boldsymbol{\pi}_h = \boldsymbol{\pi}_{h+1}$) or the predefined maximum step, which is set to 100 in our experiments. Note that the inner product $\langle 1, \boldsymbol{\pi}_h \rangle$ is the influence, i.e., the number of infected nodes. We repeat these steps with many different seed sets. For further details specific to our experiments, including the seed-set selection method, refer to Sect. 4.1

[2] Empirically, the selected s values are mostly small, reflecting the rapid convergence of diffusion models on real-world graphs due to structural properties, such as hubs, small diameters, sparse connectivity, etc.

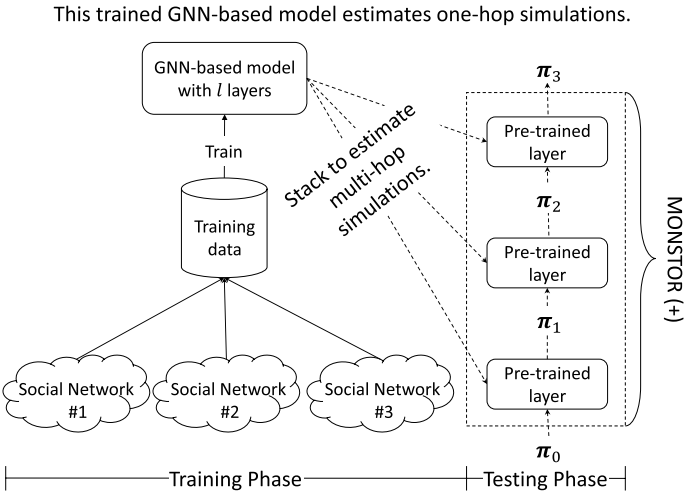


Fig. 3 The overall workflow in our approach. We train a GNN-based model and stack it s times for testing. In this example, s is set to 3, and thus MONSTOR(+) estimates $\boldsymbol{\pi}_3$ from $\boldsymbol{\pi}_0$

MONSOR(+) estimates end-to-end multi-step simulations under the considered diffusion model (IC or LT) by stacking the GNN-based model.

5. For the IE problem, we estimate the influence by $\langle 1, \boldsymbol{\pi}_s \rangle$ using the estimated $\boldsymbol{\pi}_s$. For the IM problem, we replace the MC simulation subroutine of existing IM algorithms (e.g., (Kempe et al. 2003; Goyal et al. 2011a; Zhou et al. 2013)) with MONSTOR(+).

Note that, in the training phase, we use MC simulations to obtain $\boldsymbol{\pi}_i, \boldsymbol{\pi}_{i-1}, \cdots, \boldsymbol{\pi}_{i-e}$. In the test phase, MC simulations in (potentially unseen) target networks are not needed.

3.2 Detailed design of MONSTOR (Basic Ver.)

In this subsection, we describe a basic version of our GNN-based model M and the training method for it. This basic model is stacked to compose MONSTOR. As stated earlier, M estimates $\boldsymbol{\pi}_i$ given $\boldsymbol{\pi}_{i-1}, \cdots, \boldsymbol{\pi}_{i-e}$. Specifically, M initializes the feature vector of each node v as $\mathrm{h}_v^0 := (\rho_{i-e+1}(v) - \rho_{i-e}(v), \cdots, \rho_{i-1}(v) - \rho_{i-2}(v), \rho_{i-1}(v))$, and it repeatedly computes new feature vectors of each node as follows for $1 \leq i \leq l$:

$$\mathrm{a}_v^i := \mathrm{AGG}(\{p_{(u,v)} \cdot (\mathrm{h}_u^{i-1} \mathrm{W}_1^i + \mathrm{b}_1^i) : u \in \mathrm{NEI}(v)\}), \ \forall v \in V, \tag{1}$$

$$\mathrm{h}_v^i := \mathrm{ReLU}(\mathrm{CONCAT}(\mathrm{h}_v^{i-1}, \mathrm{a}_v^i)\mathrm{W}_2^i + \mathrm{b}_2^i), \ \forall v \in V, \tag{2}$$

where $\mathrm{h}_u^i \in \mathbb{R}^{d_i}$ is the feature vector of the node u at the i-th layer; $\mathrm{NEI}(v)$ is the set of neighbors of v; AGG is MAX (i.e., the element-wise max) under the IC model or AVG (i.e., the element-wise average) under the LT model; CONCAT is the concatenation function; and $\mathrm{W}_1^i \in \mathbb{R}^{d_{i-1} \times d_{i-1}}$, $\mathrm{W}_2^i \in \mathbb{R}^{2d_{i-1} \times d_i}$, $\mathrm{b}_1^i \in \mathbb{R}^{d_{i-1}}$ and $\mathrm{b}_2^i \in \mathbb{R}^{d_i}$

are learnable parameters. Under the LT model, $p_{u,v}$ is not given, and thus we use 1 instead.

The idea of multiplying $p_{(u,v)}$ and h_u^{i-1} is inspired by the fact that activation probabilities in the IC model are multiplied following a cascade route. For instance, the probability that u_1 activates u_2 and u_2 activates u_3 is $p_{(u_1,u_2)} \cdot p_{(u_2,u_3)}$. The AVG aggregator (which involves division by the neighbor count) is used under the LT model where a node is activated when the ratio (which also involves division by the neighbor count) of its infected neighbors exceeds its threshold.

Instead of directly estimating the raw values in $\boldsymbol{\pi}_i$, our model M uses the following more effective estimation method, which is inspired by the monotonicity (see Proposition 1)[3]:

$$M(\boldsymbol{\pi}_{i-1}, \cdots, \boldsymbol{\pi}_{i-e}, \mathrm{P}\ (\text{or A}); \boldsymbol{\theta}) := \boldsymbol{\pi}_{i-1} + \mathrm{h}^l, \tag{3}$$

where $\boldsymbol{\theta}$ is the learnable parameters of M; and $\mathrm{h}^l \in \mathbb{R}^{|\mathcal{V}|}$ is the vector concatenating $\mathrm{h}_v^l \in \mathbb{R}$ (i.e., $d_l = 1$) for all $v \in V$.

Of many possible loss functions, we train our model M using the following loss function:

$$\mathcal{L} := \frac{1}{|T|} \sum_{t \in T} \frac{\|M(t; \boldsymbol{\theta}) - \boldsymbol{\pi}_i\|_1}{|\mathcal{V}|}, \tag{4}$$

where T is a training set; and $t = (\boldsymbol{\pi}_i, \boldsymbol{\pi}_{i-1}, \cdots, \boldsymbol{\pi}_{i-e}, \mathrm{P}(\text{or A})) \in T$ is a training sample.

3.3 Detailed design of MONSTOR+ (Advanced Ver.)

In this subsection, we present an advanced version of our GNN-based model M, which is stacked to compose MONSTOR+. This advanced version incorporates auxiliary structural node features and an advanced pooling function, as detailed below. Note that the overall workflow described in Sect. 3.1 and the training method in Sect. 3.2 remain the same as for MONSTOR.

3.3.1 Auxiliary node features: local cycle counts

Higher-order structures beyond edges are known to significantly affect diffusion results. Especially, Easley et al. (2010) shows a close theoretical connection between local triangle counts and diffusion outcomes. To enable M to capture such higher-order structures, we introduce local cycle counts (You et al. 2021; Dwivedi et al. 2021), which generalize local triangle counts, as auxiliary input features for M.

[3] In the conference version of this paper (Ko et al. 2020), we introduced an additional step based on a theoretical upper bound. However, further study revealed that its contribution to the final estimation accuracy is marginal. Thus, we decided to remove the step in this paper.

Given a social network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define the *local cycle* $c_r(v)$ *of length r* of a node $v \in \mathcal{V}$ as the number of cycles of length r that contain v. Computationally, it is equivalent to the v-th element of $Diag(\mathrm{A}^r)$, where A is the adjacency matrix of $\mathcal{G}$, and the function $Diag(\cdot)$ extracts the vector of diagonal entries of a given matrix. Under the IC model, as P naturally acts as weighted A, we utilize $Diag(\mathrm{P}^r)$ instead of $Diag(\mathrm{A}^r)$, thereby making $c_r(v)$ the weighted count of the local cycles of length r containing v.

The advanced GNN-based model M in MONSTOR+ uses the count of local cycles of different lengths (up to a hyperparameter q) as additional node features. That is, the feature vector of each node v is $\mathrm{h}_v^0 := (\rho_{i-e+1}(v) - \rho_{i-e}(v), \cdots, \rho_{i-1}(v) - \rho_{i-2}(v), \rho_{i-1}(v), c_2(v), c_3(v), \cdots, c_q(v))$ As stated in Sect. 4.1, we set q to 11 in our experiments. The effectiveness of these auxiliary node features is empirically confirmed in Sect. 4.4.

Naively computing $Diag(\mathrm{A}^r)$ or $Diag(\mathrm{P}^r)$ requires $O(|\mathcal{V}|^2)$ space, even when A or P is sparse; and this may limit scalability. Instead of computing $Diag(\mathrm{A}^r)$ or $Diag(\mathrm{P}^r)$ at once, we can divide it into sub-vectors and compute only one sub-vector at a time. For example, we can obtain $Diag(\mathrm{A}^r)[i : i + d]$ (i.e., the entries located between positions i and $i + d$) by computing

$$Diag(\overbrace{\mathrm{A}(\mathrm{A}(\cdots(\mathrm{A}}^{r-1}(\mathrm{A}[:, i : i + d]))), \tag{5}$$

where $\mathrm{A}[:, i : i + d]$ denotes the columns of A located between positions i and $i + d$, and the parentheses denote the order of computation. If A is sparse enough so that $||\mathrm{A}||_0 = O(|\mathcal{V}|d)$, this computation requires only $O(|\mathcal{V}|d)$ space.[4] Notably, this division-based approach does not increase the asymptotic time complexity of computing $Diag(\mathrm{A}^r)$ or $Diag(\mathrm{P}^r)$ beyond $O(r|\mathcal{V}|^3)$.

3.3.2 Advanced pooling

It is well known that the choice of a pooling function is crucial for the generalization ability of GNN-based models (Xu et al. 2020). While we carefully choose a pooling function in M (i.e., AGG in Eq. (1)) based on the characteristics of each diffusion model, our choices, which are limited to widely-used basic ones, may be sub-optimal, necessitating exploration of more expressive pooling functions.

Inspired by (Corso et al. 2020), in the GNN-based model M for MONSTOR+, we leverage an expressive pool ing function for AGG defined as:

$$\mathrm{AGG(z)} := \mathrm{CONCAT(SUM(z),\ MEAN(z),\ MAX(z),\ STD(z))}, \tag{6}$$

where SUM, MEAN, MAX, and STD are the element-wise sum, mean, max, and standard deviation functions, respectively. Note that, in the context of M, this pooling function generalizes SUM, MEAN, MAX, and STD as special cases. With this new

[4] Note that the dimensionality of $\mathrm{A}^p\mathrm{A}[:, i : i + d]$ is $|\mathcal{V}| \times d$ for any integer $0 \leq p \leq r - 1$.

design of AGG, $\mathrm{W}_2^i \in \mathbb{R}^{5d_{i-1} \times d_i}$ in Eq. (2). We set $p_{(u,v)}$ to $1/|\mathrm{NEI}(v)|$ in Eq. (1) for the LT model, where $p_{(u,v)}$ is not given. The effectiveness of this advanced pooling function is empirically confirmed in Sect. 4.4.

3.4 Time complexity analysis

Once the GNN-based model M in MONSTOR or MONSTOR+ is trained (potentially using graphs smaller than a target graph $\mathcal{G}$), estimating $\boldsymbol{\pi}_i$ in a social network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ for a seed set S takes $\mathcal{O}(lsd(|\mathcal{V}| + |\mathcal{E}|))$ time, where s is the number of stacks, l is the number of GNN layers per stack, and d is the maximum dimension of (latent) feature vectors.

For MONSTOR+, as a preprocessing step, which is executed only once for all potential seed sets, the local cycles of lengths up to q need to be counted for each node to be used as auxiliary features. As discussed in Sect. 3.3.1, this requires computing P^r or A^r for $2 \leq r \leq q$, which takes $O(q|\mathcal{V}|^3)$ time. Note that this cubic-time preprocessing limits the scalability of MONSTOR+ on large graphs, and exploring more scalable alternatives to local cycles is a promising direction for future work.

3.5 Extensions to other diffusion models

Although this paper focuses on the representative IC model (with three specific activation probability estimation methods) and the LT model, MONSTOR+ is not limited to these settings and can be extended to a broad range of diffusion models and activation probability estimation methods. In particular, thanks to its learnable nature, MONSTOR+ can be readily adapted to such models, provided that training data can be generated via simulation. As an example, in Appendix A.3, we additionally apply MONSTOR+ to the G-SIR (General markov chain Susceptible-Infected-Recovered) model (Yi et al. 2022) and demonstrate its effectiveness under the G-SIR model.

4 Experiments

In this section, we review our experiments for answering the following questions:

- **Q1. Influence estimation**: How accurately does MONSTOR+ estimate the influence of seed sets?
- **Q2. Influence maximization**: How accurate are simulation-based IM algorithms equipped with MONSTOR+ (instead of repeated MC simulations) compared to state-of-the-art competitors?
- **Q3. Ablation study**: How effective is each component of MONSTOR+?
- **Q4. Scalability**: How rapidly does the running time of MONSTOR+ increase as the size of the input graph grows?
- **Q5. Submodularity**: Is MONSTOR+ submodular as the ground-truth influence function?

Table 1 Statistics of each social network

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\sum \frac{p_{(u,v)}}{\lvert\mathcal{E}\rvert}$ in BT		$\sum \frac{p_{(u,v)}}{\lvert\mathcal{E}\rvert}$ in JI		$\sum \frac{p_{(u,v)}}{\lvert\mathcal{E}\rvert}$ in LP	
			Train	Test	Train	Test	Train	Test
Extended	11,409	58,972	0.07974	0.09194	0.03345	0.04095	0.16138	0.18371
WannaCry	35,627	169,419	0.07255	0.09466	0.02977	0.04494	0.16297	0.19785
Celebrity	15,184	56,538	0.03206	0.02787	0.00163	0.00159	0.26142	0.256

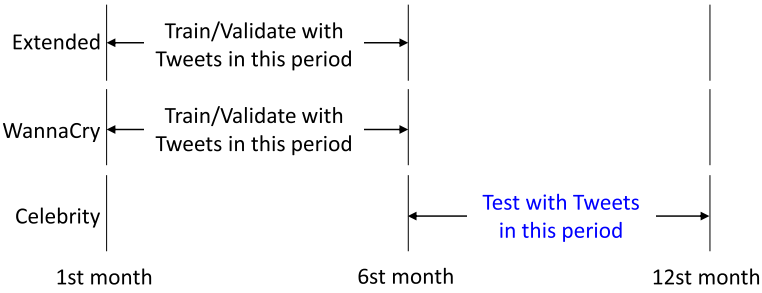


Fig. 4 Example train-test split in our experiments. We test with tweets newly posted in the test period. Note that MONSTOR and MONSTOR+ are applied to unseen social networks (i.e., the test data highlighted in blue), while existing learning approaches (Yan et al. 2019; Li et al. 2019) cannot be applied

4.1 Experimental settings

Datasets We used three real-world social networks: Extended, WannaCry, and Celebrity (Liu et al. 2019) (see Table 1 for their statistics). For Extended, we crawled more tweets and retweets in addition to those used in (Sabottke et al. 2015). In each dataset, we used online postings (and their cascade logs) during the first 50% of time for training/validation and those during the remaining 50% for testing — two sets are disjoint. Specifically, we computed the activation probability matrices (P_{BT}, P_{JI}, P_{LP}), as described in Sect. 2.1, and the adjacency matrix (A) for each of the training/validation and testing periods based on the logs. For each dataset, diffusion model, and activation probability matrix, we collected 3, 200 training tuples, 800 validation tuples, and 4, 000 testing tuples, as described in Sect. 3.1. For the tuples, we set seed sets as follows: (1) the cardinalities of the seed sets were uniformly sampled between 1 and $\frac{|\mathcal{V}|}{50}$; and (2) half of the tuples consisted of seed nodes selected uniformly at random, while the other half consisted of nodes selected with probability proportional to their degrees. In all experiments, we evaluated MONSTOR+ (and its variants) in inductive settings, unless otherwise stated. Specifically, we trained it using two out of the three social networks (e.g., Extended and WannaCry) and tested it with the remaining one, as illustrated in Fig. 4. That is, when Celebrity was used for testing, we employed the model trained using Extended and WannaCry.

Competitors We consider the following IM algorithms for comparisons: (1) Greedy (Kempe et al. 2003), UBLF (Zhou et al. 2013), and CELF (Goyal et al. 2011a) among simulation-based algorithms; (2) SSA (Nguyen et al. 2016), D-SSA (Nguyen et al. 2016), TIM (Tang et al. 2014), IMM (Tang et al. 2015), IRIE (Jung et al. 2012), PMIA (Wang et al. 2012), and SIMPATH (Goyal et al. 2011b) among non-simulation-based algorithms;, (3) C-GLIE, which is CELF equipped with GLIE (Panagopoulos et al. 2023), an inductive learning method that replaces repeated Monte Carlo simulations with a trained graph neural network, and (4) U-MON+ and C-MON+, which are UBLF and CELF equipped with MONSTOR+ replacing repeated MC simulations. Note that IRIE, PMIA, and C-GLIE are only applicable to the IC model, and SIMPATH is only applicable to the LT model. Also note that the outputs of Greedy, UBLF, and CELF are the same. Among them, UBLF is the fastest, followed by CELF and Greedy. U-MON+ and C-MON+ also output the same seed users. UBLF and U-MON+ can be used only when activation probabilities satisfy a certain property (Zhou et al. 2013). Due to this reason, for the IC model with LP and the LT model, only C-MON+ is applicable.

Hyperparameters

For MONSTOR+, we set $e = 4$, $l = 3$, $q = 11$, $d_1 = \cdots = d_{l-1} = 16$ after some preliminary studies. At each t-th epoch, we set the learning rate to $10^{-4} \cdot t$ if $t \leq 10$, and $10^{-2}/t$ otherwise. We selected the s value that minimizes RMSE on the validation set among $\{2, \cdots, 20\}$. For (D-)SSA, we set $\epsilon = 0.1$ and $\delta = 1/|\mathcal{V}|$ as in (Nguyen et al. 2016). For TIM (Tang et al. 2014) and IMM (Tang et al. 2015), we set $\epsilon = 0.1$. For IRIE and PMIA, we followed the settings in (Jung et al. 2012; Wang et al. 2012). For SIMPATH, we set the cut-off threshold to 0.001 and the number of items for Look Ahead Optimization to 4, as in (Goyal et al. 2011b).

4.2 Q1. Influence estimation (IE)

In this section, we demonstrate that MONSTOR+ accurately estimates the influence for given seed sets for the influence estimation (IE) problem, which naturally leads to high-quality seed nodes for the IM problem. As obtaining exact ground-truth influence is intractable, we used the mean influence from 10,000 Monte Carlo (MC) simulations as the ground truth.

4.2.1 Pearson and rank correlation coefficients

We begin by showing that MONSTOR+ produces estimates with near-perfect correlation to the ground-truth influence, significantly outperforming GLIE (Panagopoulos et al. 2023), a graph neural network trained to estimate influence under the IC model. To this end, for each test seed set described in Sect. 4.1, we compared the estimated influences of MONSTOR+ and GLIE. To measure the similarity between ground-truth and estimated influences overall test seed sets, we used Pearson's correlation coefficients and Spearman's Rank correlation coefficients. As seen in Table 2a, the ground-truth influences and the estimated influences of MONSTOR+ for test seed sets are highly correlated, and both correlation coefficients were close to 1.0. More-

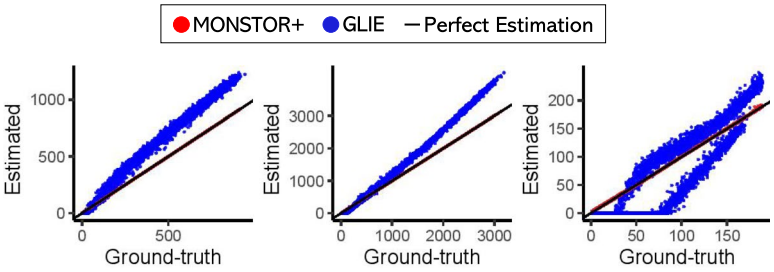
Table 2 The accuracy (specifically, the correlation between estimated and ground-truth influence) of MONSTOR+ and GLIE on the influence estimation problem. GLIE is applicable only to the IC model. Note that MONSOTR+ consistently outperforms GLIE

(a) MONSTOR+

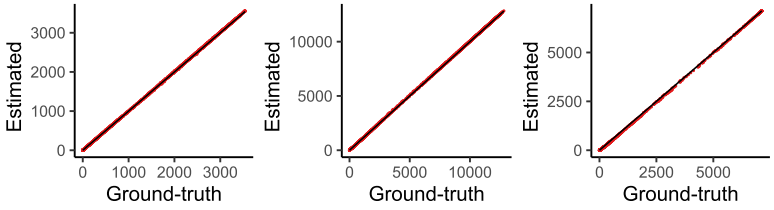
	IC Model						LT Model	
	Pearson Correlation			Rank Correlation			Pearson	Rank
Dataset	BT	JI	LP	BT	JI	LP	Correlation	Correlation
Extended	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.989
WannaCry	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.955
Celebrity	0.995	1.000	1.000	0.998	1.000	0.999	1.000	0.995

(b) GLIE

Dataset	IC Model					
	Pearson Correlation			Rank Correlation		
	BT	JI	LP	BT	JI	LP
Extended	0.997	0.992	0.998	0.996	0.986	0.993
WannaCry	0.999	0.989	0.997	0.998	0.983	0.994
Celebrity	0.872	0.745	0.989	0.884	0.767	0.937

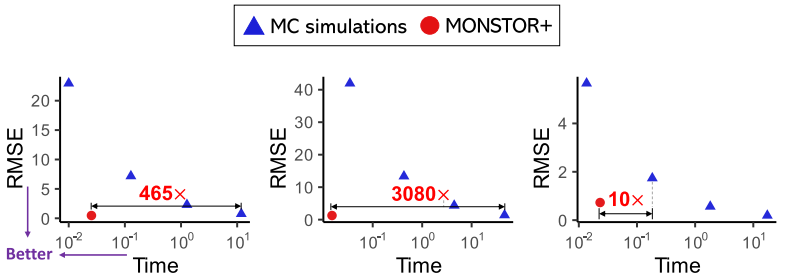


(a) IC model with BT: Extended (left), WannaCry (center), and Celebrity (right)

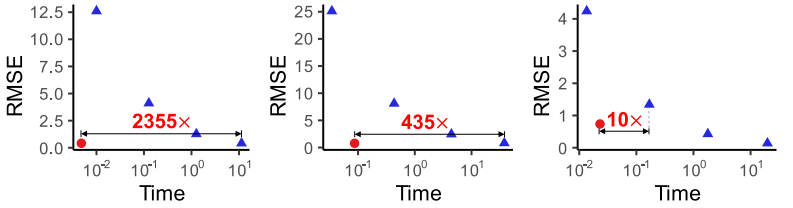


(b) LT model: Extended (left), WannaCry (center), and Celebrity (right)

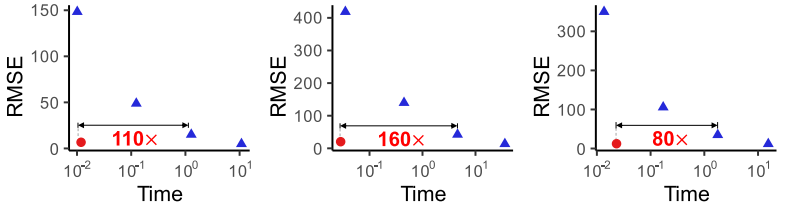
Fig. 5 Scatter plots showing ground-truth influence versus estimates from MONSTOR+ or GLIE on test datasets unseen during training. Red and blue dots represent estimates from MONSTOR+ and GLIE, respectively, while black lines indicate perfect estimation. GLIE is applicable only to the IC model. Note that estimates from MONSTOR+ lie closer to the perfect estimation line than those from GLIE. Refer to Appendix A.1 for the results under other diffusion models



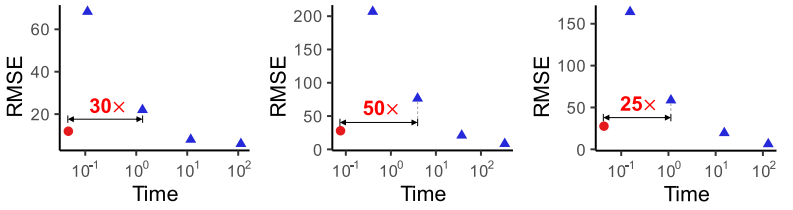
(a) IC model with BT: Extended (left), WannaCry (center), and Celebrity (right)



(b) IC model with JI: Extended (left), WannaCry (center), and Celebrity (right)



(c) IC model with LP: Extended (left), WannaCry (center), and Celebrity (right)

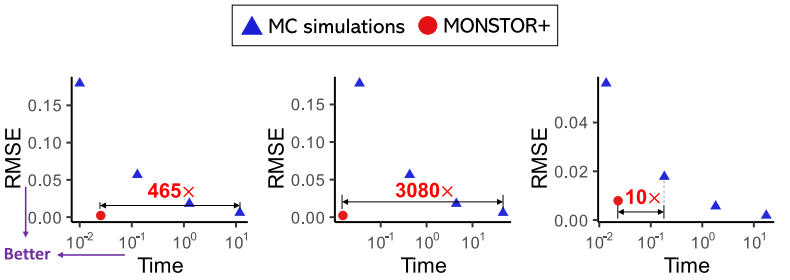


(d) LT model: Extended (left), WannaCry (center), and Celebrity (right)

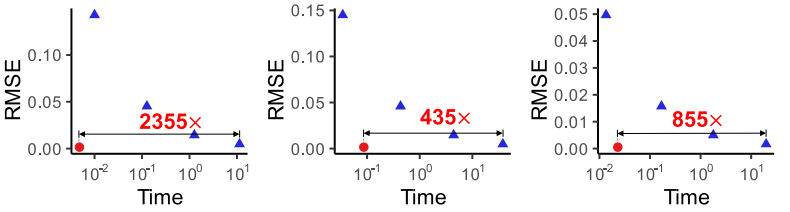
Fig. 6 Root mean square error (RMSE) of the influences estimated by MONSTOR+ and repeated MC simulations (with varying numbers of repetitions) in test datasets unseen during training. The further toward the lower-left corner a method is positioned in the plot, the faster and more accurate it is.

over, by comparing Table 2a and Table 2b, we see that MONSTOR+ consistently yielded more accurate influence estimates than GLIE.

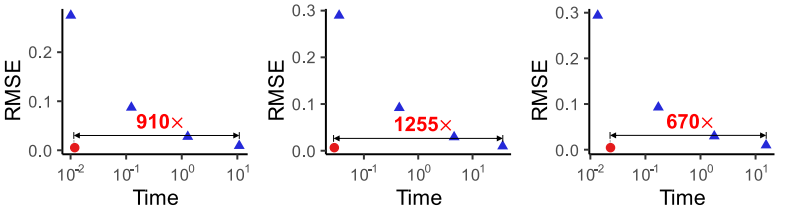
We further illustrate this effectiveness in Fig. 5, which presents scatter plots comparing ground-truth and estimated influences on test datasets unseen during training (refer to Appendix A.1 for the results under more diffusion models). Note that



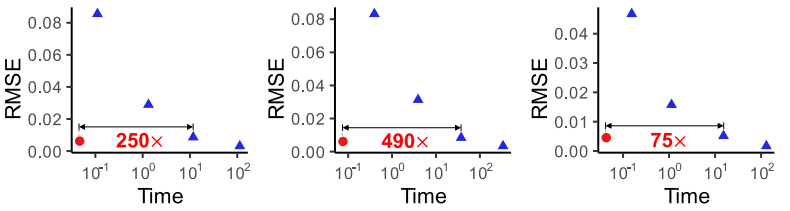
(a) IC model with BT: Extended (left), WannaCry (center), and Celebrity (right)



(b) IC model with JI: Extended (left), WannaCry (center), and Celebrity (right)



(c) IC model with LP: Extended (left), WannaCry (center), and Celebrity (right)



(d) LT model: Extended (left), WannaCry (center), and Celebrity (right)

Fig. 7 Root mean square error (RMSE) of the infection probabilities estimated by MONSTOR+ and repeated MC simulations (with varying numbers of repetitions) in test datasets unseen during training. The closer a method is to the lower-left corner, the faster and more accurate it is.

estimated influences from MONSTOR+ lie closer to the perfect estimation line than those from GLIE.

Table 3 The performance (i.e., the influence of output seeds) of IM methods under the IC model with BT. We mark the best performances in **bold**

(a) Extended				
	$k = 10$	50	100	Average rank
Target Influence	480.8 ± 29.7	968.3 ± 28.8	1219.9 ± 28.3	–
U-MON+ (Proposed)	**481.1 ± 31.1**	**968.5 ± 28.6**	**1223.2 ± 28.8**	**1**
D-SSA	468.3 ± 30.9	960.3 ± 29.4	1203.9 ± 29.0	5.33
SSA	466.7 ± 31.3	939.9 ± 28.4	1167.8 ± 28.87	7
TIM	**481.1 ± 31.1**	**968.5 ± 28.6**	**1223.2 ± 28.5**	**1**
IMM	**481.1 ± 31.1**	968.2 ± 29.0	1221.3 ± 28.7	2.67
IRIE	479.6 ± 30.6	966.0 ± 28.9	1221.7 ± 29.4	3.67
PMIA	473.3 ± 30.3	960.1 ± 27.5	1199.4 ± 26.9	5.67
C-GLIE	423.0 ± 29.4	811.9 ± 26.0	994.6 ± 25.3	8
(b) WannaCry				
	$k = 10$	50	100	Average rank
Target Influence	990.4 ± 67.5	2124.5 ± 59.2	2747.8 ± 55.7	–
U-MON+ (Proposed)	**991.1 ± 68.0**	**2123.7 ± 59.1**	**2753.3 ± 55.7**	**1**
D-SSA	986.2 ± 67.5	2108.1 ± 59.9	2733.7 ± 56.3	5.67
SSA	983.9 ± 67.8	2093.5 ± 65.6	2666.1 ± 57.4	7
TIM	**991.1 ± 68.0**	**2123.7 ± 59.1**	**2753.3 ± 55.7**	**1**
IMM	989.0 ± 71.6	2122.0 ± 59.2	2752.4 ± 55.8	3.33
IRIE	986.6 ± 63.8	2118.5 ± 59.8	2751.6 ± 56.8	4.33
PMIA	989.3 ± 59.6	2106.9 ± 55.3	2739.8 ± 52.6	3.67
C-GLIE	825.0 ± 73.0	1394.9 ± 68.8	2181.2 ± 55.3	8
(c) Celebrity				
	$k = 10$	50	100	Average rank
Target Influence	51.0 ± 6.6	104.0 ± 6.5	154.0 ± 6.6	–
U-MON+ (Proposed)	52.8 ± 6.9	**105.1 ± 6.6**	**155.0 ± 6.6**	**1.33**
D-SSA	50.8 ± 7.1	102.4 ± 6.7	152.2 ± 6.7	5.67
SSA	50.9 ± 7.1	101.6 ± 6.7	152.1 ± 6.7	6
TIM	**52.9 ± 6.8**	105.0 ± 6.8	154.9 ± 6.6	1.67
IMM	52.8 ± 6.8	104.6 ± 6.6	154.7 ± 6.6	2.67
IRIE	51.7 ± 6.7	103.0 ± 6.6	153.0 ± 6.6	4
PMIA	51.7 ± 6.7	100.0 ± 6.6	152.1 ± 6.6	5.67
C-GLIE	31.5 ± 6.1	88.6 ± 7.9	142.0 ± 6.6	8

4.2.2 Comparison with Monte Carlo (MC) simulations

We present a numerical comparison between MONSTOR+ and repeated MC simulations, which are widely used for influence estimation. To this end, we measure the root mean square error (RMSE) of the influence (ϕ in Problem 1) and the infection probability of each node (π in Problem 1) estimated by MONSTOR+ and MC simulations with different numbers of repetitions (specifically, 10, 100, 1,000, 10,000 for the LT model; and 1, 10, 100, 1,000 for the IC model).

As seen in Figs. 6 and 7, in most cases, MONSTOR+ was significantly faster, typically between 5 to 3000 times, compared to MC simulations, for similar levels of estimation error. Similar findings were observed when evaluating the mean absolute percentage error (MAPE), as detailed in Appendix A.1. These results demonstrate the capability of MONSTOR+ to successfully replace MC simulations, offering a substantial speed-up.

4.3 Q2. Influence maximization (IM)

We demonstrate the effectiveness of MONSTOR+ for the influence maximization (IM) problem (Problem 2). To this end, we measured the performance (i.e., the influence of output seeds) of various IM methods (see Sect. 4.1) under both the IC and LT models. Recall that U-MON+ and C-MON+ are UBLF and CELF equipped with MONSTOR+, which replaces their repeated MC simulations. Between them, we employed U-MON+ whenever applicable and used C-MON+ otherwise (see Sect. 4.1 for details). The results can be found in Tables 4-6, where the target influences refer to those obtained by the greedy algorithm with repeated MC simulations (10, 000 in our experiments), which MONSTOR+ and many baselines aim to approximate using machine learning, sketching, and other techniques. Even though we used CELF++ (Goyal et al. 2011a), which is a more efficient variant of the greedy algorithm that produces identical output, for obtaining the target influences, it still required at least two orders of magnitude more time than the compared methods.

As seen in Table 4, under the IC model with JI, U-MON+ performed best in all cases. Similarly, as seen in Tables 3 and 6, U-MON+ and C-MON+ outperformed other methods in almost all cases, with only one or two exceptions, under the IC model with BT and the LT model, respectively. U-MON+ also performed overall best under the IC model with LP as shown in Table 5. In total, U-MON+ and C-MON+ (i.e., IM algorithms equipped with MONSTOR+) were most accurate in 81.5% and 77.8% of IM use cases under the IC and LT models, respectively. Moreover, in most cases, their performances were close to or sometimes even better than the target performances.

4.4 Q3. Ablation study

We demonstrate the importance of two key enhancements in MONSTOR+: the auxiliary structural node features (discussed in Sect. 3.3.1) and the advanced pooling function (explained in Sect. 3.3.2). To evaluate their impact, we measured the root mean square error (RMSE) of the influence and infection probabilities estimated by

four variations: (1) MONSTOR+ (with all enhancements), (2) MONSTOR+ without the auxiliary structural node features, (3) MONSTOR+ without the advanced pooling function, and (4) MONSTOR (MONSTOR+ without the auxiliary structural node features and the advanced pooling function). We also evaluated their performance on the IM problem across different diffusion models.

As shown in Tables 7 and 8, both enhancements improved the estimation accuracy of MONSTOR+. Tables 9 and 10 further demonstrate that these enhancements also lead to improved performance on the IM problem (also refer to Appendix A.2 for additional results). Specifically, the auxiliary structural node features played a more significant role under the LT model, whereas the advanced pooling function had a greater impact under the IC model for both IE and IM problems. The impact of the auxiliary structural node features was relatively minor in JI cases, where activation probabilities are low, resulting in smaller weights for local cycles. The effectiveness of both enhancements was also evident in terms of mean absolute percentage error (MAPE), as detailed in Appendix A.2.

4.5 Q4. Scalability

We present our analysis of the scalability of MONSTOR+. For this analysis, we generated realistic graphs of various sizes using the R-MAT generator (Chakrabarti et al. 2004) with parameters $a = 0.7$ and $b = c = d = 0.1$. The number of edges in the generated graphs ranged from 2^{20} to 2^{24}, and for every graph, we set the number of nodes to 20% of the number of edges. We used the weighted cascade model (Kempe et al. 2003) to determine the activation probability of each edge in the generated graphs. To reduce memory requirements during the pre-processing of MONSTOR+ (i.e., computing the auxiliary structural node features), as discussed in Sect. 3.3.1, we partitioned the columns of P and A into blocks.[5] For estimation time, we divided the runtime for influence estimations for 1, 000 different seed sets by the number of seed sets. For each seed set, we first chose its size uniformly at random from 1 to 10% of $|\mathcal{V}|$ and then chose seed nodes uniformly at random. Moreover, since the runtime scaled linearly with the number of stacked GNNs (i.e., s), we measured the estimation time per stacked GNN.

In Table 11, we present the runtime of (1) pre-processing of MONSTOR+ and (2) influence estimation by MONSTOR and MONSTOR+ on graphs of varying sizes. For MONSTOR+, as expected from the theoretical analysis in Sect. 3.4, the runtime for pre-processing, which is executed only once, increased super-linearly with respect to the number of edges in the input graph. The runtime for influence estimation by MONSTOR and MONSTOR+, which is performed repeatedly for each seed set, exhibited near-linear scalability.

4.6 Q5. Submodularity

It is well known that the influence maximization is a submodular maximization problem, and many IM algorithms exploit the submodularity of the ground-truth influence

[5] Specifically, each block consisted of $10,000 \times 2^{20}/|\mathcal{E}|$ columns.

Table 4 The performance (i.e., the influence of output seeds) of IM methods under the IC model with JI. We mark the best performances in **bold**

(a) Extended				
	$k = 10$	50	100	Average rank
Target Influence	244.5 ± 16.3	529.3 ± 20.7	705.4 ± 22.3	–
U-MON+ (Proposed)	**244.4 ± 16.3**	**529.4 ± 20.7**	**707.2 ± 22.5**	**1**
D-SSA	243.8 ± 16.4	524.7 ± 20.7	697.6 ± 22.4	6
SSA	242.5 ± 16.4	520.5 ± 20.8	690.6 ± 22.4	7
TIM	**244.4 ± 16.3**	529.3 ± 20.8	**707.2 ± 22.5**	1.33
IMM	**244.4 ± 16.3**	529.3 ± 20.8	707.1 ± 22.4	2.33
IRIE	**244.4 ± 16.3**	529.3 ± 20.7	**707.2 ± 22.4**	1.33
PMIA	**244.4 ± 16.3**	529.2 ± 20.6	705.4 ± 22.3	3.67
C-GLIE	131.4 ± 12.1	348.6 ± 20.4	553.7 ± 22.5	8
(b) WannaCry				
	$k = 10$	50	100	Average rank
Target Influence	534.3 ± 24.3	1238.6 ± 34.6	1646.9 ± 36.9	–
U-MON+ (Proposed)	**534.1 ± 24.3**	**1239.0 ± 34.3**	**1648.2 ± 37.2**	**1**
D-SSA	531.4 ± 24.8	1229.6 ± 34.5	1632.5 ± 37.3	6
SSA	527.1 ± 25.2	1196.1 ± 34.5	1583.2 ± 37.0	7
TIM	**534.1 ± 24.3**	1238.9 ± 34.1	1648.1 ± 37.1	2
IMM	533.1 ± 25.1	1238.9 ± 34.3	1647.7 ± 37.2	4
IRIE	**534.1 ± 24.3**	**1239.0 ± 34.2**	1647.9 ± 37.6	1.67
PMIA	**534.1 ± 24.3**	1238.9 ± 34.3	1646.8 ± 36.8	3
C-GLIE	125.0 ± 16.7	508.6 ± 31.2	1255.8 ± 35.9	8
(c) Celebrity				
	$k = 10$	50	100	Average rank
Target influence	43.2 ± 5.7	89.9 ± 6.3	139.8 ± 6.2	–
U-MON+ (Proposed)	**43.7 ± 5.8**	**90.4 ± 6.3**	**140.5± 6.3**	**1**
D-SSA	**43.7 ± 5.8**	89.7 ± 6.2	139.9 ± 6.2	5
SSA	**43.7 ± 5.8**	90.0 ± 6.3	140.0 ± 6.3	4.67
TIM	**43.7 ± 5.8**	90.3 ± 6.3	140.1 ± 6.3	3
IMM	**43.7 ± 5.8**	90.1 ± 6.2	140.1 ± 6.3	3.33
IRIE	**43.7 ± 5.8**	**90.4 ± 6.3**	140.3 ± 6.3	1.33
PMIA	42.7 ± 5.7	**90.4 ± 6.3**	140.3 ± 6.3	2.33
C-GLIE	21.4 ± 3.4	72.7 ± 5.0	138.4 ± 6.1	8

Table 5 The performance (i.e., the influence of output seeds) of IM methods under the IC model with LP. We mark the best performances in **bold**

(a) Extended

	$k = 10$	50	100	Average rank
Target influence	1854.9 ± 135.6	2875.5 ± 60.5	3262.8 ± 45.7	–
C-MON+ (Proposed)	1852.2 ± 119.9	**2876.8 ± 60.9**	3265.6 ± 48.0	**1.67**
D-SSA	1849.0 ± 128.0	2865.6 ± 66.1	3236.3 ± 51.3	4.33
SSA	1848.0 ± 127.5	2861.5 ± 61.8	3230.6 ± 48.4	5.33
TIM	1852.2 ± 119.9	2876.5 ± 61.5	**3267.0 ± 47.0**	**1.67**
IMM	**1852.3 ± 135.9**	2875.5 ± 61.1	3264.4 ± 47.8	2.33
IRIE	1816.1 ± 135.4	2829.2 ± 73.0	3201.2 ± 57.4	6.67
PMIA	1829.9 ± 109.2	2828.7 ± 54.8	3243.1 ± 42.2	5.67
C-GLIE	1795.0 ± 121.4	2764.5 ± 58.1	2987.0 ± 44.9	8

(b) WannaCry

	$k = 10$	50	100	Average rank
Target influence	5264.7 ± 389.3	7877.3 ± 200.3	9085.3 ± 136.2	–
C-MON+ (Proposed)	**5264.9 ± 391.6**	**7888.4 ± 197.3**	9106.2 ± 131.3	**1.33**
D-SSA	5247.4 ± 393.6	7857.2 ± 202.1	8983.1 ± 150.9	4.33
SSA	5244.1 ± 400.1	7819.3 ± 205.9	8996.1 ± 145.3	4.67
TIM	**5264.9 ± 391.6**	7887.6 ± 197.2	**9106.6 ± 131.3**	**1.33**
IMM	**5264.9 ± 391.6**	7887.4 ± 203.8	9102.8 ± 134.8	2.33
IRIE	5112.4 ± 467.7	7714.0 ± 257.3	8841.1 ± 206.0	7
PMIA	5196.1 ± 352.8	7807.4 ± 167.6	8981.3 ± 115.3	6
C-GLIE	5037.1 ± 442.4	7579.1 ± 173.9	8594.5 ± 122.2	8

(c) Celebrity

	$k = 10$	50	100	Average rank
Target Influence	5508.5 ± 42.4	5607.7 ± 33.3	5640.9 ± 32.8	–
C-MON+ (Proposed)	**5509.0 ± 42.5**	**5617.2 ± 33.2**	5667.1 ± 33.3	**1.33**
D-SSA	5508.8 ± 42.5	5609.9 ± 33.2	5637.6 ± 33.1	6
SSA	5508.9 ± 42.6	5602.4 ± 33.4	5638.3 ± 33.4	6.33
TIM	**5509.0 ± 42.5**	5617.1 ± 33.2	**5667.2 ± 33.3**	1.67
IMM	**5509.0 ± 42.5**	5617.1 ± 33.1	5666.4 ± 33.2	2.67
IRIE	**5509.0 ± 42.5**	**5617.2 ± 33.2**	5667.1 ± 33.3	**1.33**
PMIA	**5509.0 ± 42.5**	5603.8 ± 33.2	5630.8 ± 33.1	5
C-GLIE	5443.1 ± 67.7	5609.4 ± 33.3	5621.6 ± 33.1	7.33

function. Therefore, it is crucial to show that MONSTOR+ has the same characteristic for claiming that IM algorithms based on the submodularity work properly when being integrated with MONSTOR+.

In this section, we review our experiments for testing the empirical submodularity of MONSTOR+. To this end, we used 5, 000 test seed sets that were not used for training. For each seed set, we chose its size uniformly at random from 1 to 10% of $|\mathcal{V}|$ and then chose seed nodes uniformly at random. Using each pair S and T of the seed sets, we tested whether the following submodularity condition is met:

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T). \tag{7}$$

Table 6 The performance (i.e., the influence of output seeds) of IM methods under the LT model. We mark the best performances in **bold**

(a) Extended				
	$k = 10$	50	100	Average rank
Target Influence	2273.8 ± 268.5	3763.6 ± 115.9	4332.4 ± 70.1	–
C-MON+ (Proposed)	**2271.9 ± 267.9**	**3782.3 ± 119.2**	**4346.3 ± 69.3**	**1**
D-SSA	2259.5 ± 2745	3750.0 ± 121.5	4306.9 ± 76.4	4.67
SSA	2263.6 ± 274.0	3764.8 ± 111.0	4334.2 ± 74.8	3.33
TIM	**2271.9 ± 267.9**	**3782.3 ± 110.5**	4345.9 ± 68.8	**1.33**
IMM	**2271.9 ± 267.9**	3752.1 ± 115.9	4289.9 ± 74.7	3.33
SIMPATH	2235.7 ± 225.6	3714.9 ± 103.1	4262.8 ± 61.5	6
(b) WannaCry				
	$k = 10$	50	100	Average rank
Target Influence	7448.6 ± 659.8	11008.2 ± 315.4	12606.6 ± 222.3	–
C-MON+ (Proposed)	**7452.4 ± 660.7**	**11011.6 ± 316.8**	12603.0 ± 226.9	**1.33**
D-SSA	7429.1 ± 667.3	10960.3 ± 303.1	12520.5 ± 232.0	3.67
SSA	7430.5 ± 669.5	10970.8 ± 362.6	12494.6 ± 247.8	3.33
TIM	**7452.4 ± 660.7**	11010.5 ± 313.2	**12606.6 ± 222.5**	**1.33**
IMM	7397.5 ± 706.7	10934.8 ± 364.0	12481.3 ± 244.5	5
SIMPATH	7256.1 ± 739.6	10742.6 ± 336.2	12367.5 ± 226.6	6
(c) Celebrity				
	$k = 10$	50	100	Average rank
Target Influence	6944.2 ± 116.1	7122.9 ± 1.5	7172.9 ± 1.5	–
C-MON+ (Proposed)	6944.7 ± 115.2	**7123.8 ± 7.6**	**7173.9 ± 0.8**	**1.33**
D-SSA	6943.1 ± 108.8	7123.3 ± 1.9	7173.3 ± 1.5	3.67
SSA	6944.6 ± 107.2	7122.5 ± 1.9	7172.5 ± 2.2	4
TIM	6944.7 ± 115.2	**7123.8 ± 7.5**	**7173.9 ± 0.8**	**1.33**
IMM	**6944.8 ± 114.9**	7120.5 ± 29.3	7172.4 ± 1.5	3.67
SIMPATH	6666.8 ± 314.2	6856.0 ± 272.5	6968.5 ± 242.8	6

In Table 12, we show the ratio of the pairs where the above submodularity condition is met. The submodularity condition held for 99% or higher of the pairs in the Extended and WannaCry datasets regardless of the underlying diffusion models. However, in Celebrity (especially the IC model with JI), the ratio was much lower. One possible reason for this is that, in Celebrity, activation probabilities with JI are relatively small compared to BT and LP, as shown in Table 1.

For each pair S and T where the submodularity condition was not met, we measured the mean absolute percentage error (MAPE) as follows:

$$\frac{f(S \cup T) + f(S \cap T) - f(S) - f(T)}{f(S \cup T) + f(S \cap T)}. \tag{8}$$

Table 7 Ablation study. Root mean square error (RMSE) of the influences estimated by MONSTOR+ and its variants with missing components in test datasets unseen during training

	Auxiliary node feature	Advanced pooling	IC model			LT model	Average rank
			BT	JI	LP		
(a) Extended							
(1)	✓	✓	**0.47**	0.42	**6.68**	**8.31**	**1.25**
(2)		✓	0.49	**0.20**	12.26	31.92	2
(3)	✓		4.50	1.63	64.92	16.18	3.25
(4)			6.85	1.28	27.68	47.80	3.5
(b) WannaCry							
	Auxiliary node feature	Advanced pooling	IC model			LT model	Average rank
			BT	JI	LP		
(1)	✓	✓	**1.28**	0.78	**20.32**	**27.93**	**1.25**
(2)		✓	1.57	**0.52**	31.78	141.12	2.25
(3)	✓		8.75	3.47	98.76	63.55	3.25
(4)			19.27	2.94	88.50	123.65	3.25
(c) Celebrity							
	Auxiliary node feature	Advanced pooling	IC model			LT model	Average rank
			BT	JI	LP		
(1)	✓	✓	**0.73**	0.74	**12.32**	**27.65**	**1.5**
(2)		✓	7.13	**0.09**	46.28	86.35	2.25
(3)	✓		11.07	0.30	733.02	42.53	3
(4)			5.90	0.87	109.61	100.14	3.25

Table 8 Ablation study. Root mean square error (RMSE) of the infection probabilities (of individual nodes) estimated by MONSTOR+ and its variants with missing components in test datasets unseen during training

	Auxiliarynode feature	Advanced pooling	IC model			LT model	Average rank
			BT	JI	LP		
(a) Extended							
(1)	✓	✓	**0.0022**	**0.0015**	**0.0053**	**0.0077**	**1**
(2)		✓	**0.0022**	**0.0015**	0.0095	0.0200	1.75
(3)	✓		0.0088	0.0027	0.0470	0.0087	3.25
(4)			0.0084	0.0032	0.0300	0.0253	3.5
(b) WannaCry							
	Auxiliary node feature	Advanced pooling	IC model			LT model	Average rank
			BT	JI	LP		
(1)	✓	✓	**0.0022**	**0.0015**	**0.0066**	**0.0061**	**1**
(2)		✓	**0.0022**	**0.0015**	0.0087	0.0205	1.75
(3)	✓		0.0063	0.0029	0.0317	0.0072	2.75
(4)			0.0081	0.0033	0.0325	0.0229	4
(c) Celebrity							
	Auxiliary node feature	Advanced pooling	IC model			LT model	Average rank
			BT	JI	LP		
(1)	✓	✓	0.0079	**0.0005**	**0.0044**	**0.0045**	**1.75**
(2)		✓	**0.0024**	**0.0005**	0.0092	0.0150	**1.75**
(3)	✓		0.0056	0.0006	0.1235	0.0062	3
(4)			0.0055	0.0006	0.0602	0.0152	3

Table 9 Ablation study. The influence maximization performance (i.e., the influence of output seeds) of MONSTOR+ and its variants with missing components under the IC model with LP in test datasets unseen during training. Refer to Appendix A.2 for the results under the IC model with BT and JI

(a) Extended						
	Auxiliary node feature	Advanced pooling	$k = 10$	50	100	Average rank
(1)	✓	✓	**1852.2 ± 119.9**	**2876.8 ± 60.9**	**3265.6 ± 48.0**	**1**
(2)		✓	1849.5 ± 123.4	2875.6 ± 61.2	3265.5 ± 47.5	2
(3)	✓		1849.3 ± 123.5	2866.0 ± 63.4	3243.8 ± 51.2	3.67
(4)			1849.3 ± 123.3	2868.0 ± 64.2	3251.2 ± 50.1	3
(b) WannaCry						
	Auxiliary node feature	Advanced pooling	$k = 10$	50	100	Average rank
(1)	✓	✓	5264.9 ± 391.6	**7888.4 ± 197.3**	**9106.2 ± 131.3**	**1.33**
(2)		✓	**5266.1 ± 391.4**	7884.3 ± 192.7	9103.2 ± 134.3	1.67
(3)	✓		5241.4 ± 401.1	7855.4 ± 205.8	9075.0 ± 147.9	3.67
(4)			5266.3 ± 391.1	7880.0 ± 197.4	9060.5 ± 135.5	3.33
	Auxiliary node feature	Advanced pooling	$k = 10$	50	100	Average rank
(1)	✓	✓	**5509.0 ± 42.5**	**5617.2 ± 33.2**	**5667.1 ± 33.3**	**1**
(2)		✓	5508.8 ± 42.6	5617.1 ± 33.2	5667.0 ± 33.2	2
(3)	✓		5443.6 ± 67.7	5606.2 ± 33.2	5656.3 ± 33.2	3.67
(4)			5365.7 ± 95.9	5610.4 ± 33.2	5660.4 ± 33.2	3.33

As shown in Table 13, the error (i.e., $f(S \cup T) + f(S \cap T) - f(S) - f(T)$) was marginal compared to the actual influence (i.e, $f(S \cup T) + f(S \cap T)$). All these experiment results support that influence estimation by MONSTOR+ can be considered as submodular in practice.

5 Related work

Independent cascade (IC) and *linear threshold* (LT) are the two most extensively studied diffusion models. In the IC model, once a node u is infected, it attempts once to infect each direct neighbor v independently with probability $p_{(u,v)}$. In the LT model, a node v is infected if a sufficient number of its direct neighbors (larger than a threshold) are infected. Numerous successful real-world studies are based on the IC (Yadav et al. 2016; Wilder and Vorobeychik 2018) and LT (He et al. 2012; Ou et al. 2016) models.

Influence maximization (i.e., to find a certain number of seed nodes that maximize the diffusion through a social network) is an NP-hard problem for both diffusion models (Kempe et al. 2003). Numerous methods have been proposed for influence maximization. Due to its NP-hardness, all these methods are approximate and do not guarantee the optimality of the output seed sets. They can be categorized into the following three types: (1) simulation-based, (2) sketch-based, and (3) proxy-based methods. In the simulation-based methods, MC simulations are explicitly repeated to estimate the influence of seed sets (Goyal et al. 2011a; Zhou et al. 2013). These meth-

Table 10 Ablation study. The influence maximization performance (i.e., the influence of output seeds) of MONSTOR+ and its variants with missing components under the LT model in test datasets unseen during training

	Auxiliary node feature	Advanced pooling	$k = 10$	50	100	Average rank
(a) Extended						
(1)	✓	✓	**2271.9 ± 267.9**	**3782.3 ± 119.2**	**4346.3 ± 69.3**	**1**
(2)		✓	2252.6 ± 255.9	3759.9 ± 121.0	4334.2 ± 83.1	3
(3)	✓		2271.5 ± 267.7	3782.2 ± 119.0	4346.1 ± 69.2	2
(4)			2177.7 ± 250.8	3753.6 ± 130.4	4323.4 ± 82.4	4
(b) WannaCry						
	Auxiliary node feature	Advanced pooling	$k = 10$	50	100	Average rank
(1)	✓	✓	**7452.4 ± 660.7**	**11011.6 ± 316.8**	12603.0 ± 226.9	**1.33**
(2)		✓	7422.9 ± 620.0	10991.2 ± 308.6	12572.8 ± 209.5	3.33
(3)	✓		7450.6 ± 659.3	11008.9 ± 314.8	**12604.8 ± 217.1**	1.67
(4)			7424.0 ± 619.3	10974.1 ± 292.5	12560.5 ± 206.0	3.67
(c) Celebrity						
	Auxiliary node feature	Advanced pooling	$k = 10$	50	100	Average rank
(1)	✓	✓	6944.7 ± 115.2	**7123.8 ± 7.6**	**7173.9 ± 0.8**	**1.33**
(2)		✓	6944.4 ± 97.6	**7123.8 ± 0.8**	7173.8 ± 0.8	2
(3)	✓		**6944.8 ± 97.5**	**7123.8 ± 0.8**	7173.8 ± 0.8	**1.33**
(4)			6944.4 ± 97.5	**7123.8 ± 0.8**	7173.8 ± 0.8	2

Table 11 The runtime of pre-processing (performed only once) and estimation (performed repeatedly for each seed set) by MONSTOR(+) in graphs with varying numbers of edges

(a) Pre-processing (performed only once) for MONSTOR+					
$\lvert\mathcal{E}\rvert$	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
Elapsed time (seconds)	408.68	1629.10	5584.23	25669.33	135908.99
(b) Influence estimation for MONSTOR+ per each seed set					
$\lvert\mathcal{E}\rvert$	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
Estimation time (milliseconds)	56.87	104.30	179.06	311.59	638.63
(c) Influence estimation for MONSTOR per each seed set					
$\lvert\mathcal{E}\rvert$	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
Estimation time (milliseconds)	32.3	58.5	100.0	137.7	222.5

ods focus on pruning unnecessary (redundant) simulations to minimize the required number of simulations. Among strong sketch-based methods, SSA (Nguyen et al.

Table 12 The ratio of cases where the submodularity holds in influence estimates provided by MONSTOR+

Dataset	IC model			LT model
	BT	JI	LP	
Extended	0.997	0.992	0.999	0.996
WannaCry	1.000	0.998	0.997	0.992
Celebrity	0.904	0.670	0.959	0.871

Table 13 Mean absolute percentage error (MAPE) of MONSTOR+ for the cases where submodularity does not hold

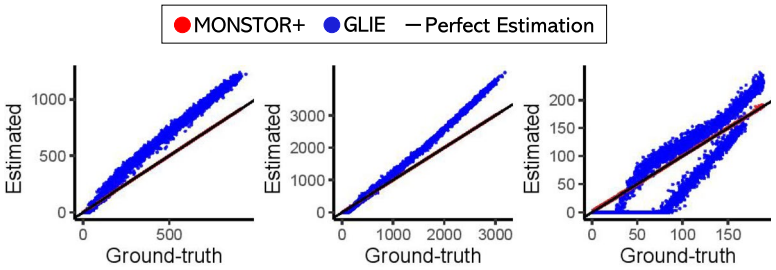
Dataset	IC model			LT model
	BT	JI	LP	
Extended	7.26e−05	7.83e−05	0.0017	0.0006
WannaCry	1.90e−09	2.14e−05	0.0048	0.0050
Celebrity	0.0012	5.18e−05	0.0073	0.0239

2016), D-SSA (Nguyen et al. 2016), TIM (Tang et al. 2014), and IMM (Tang et al. 2015) are applicable to both IC and LT models. Among strong proxy-based methods, IRIE (Jung et al. 2012) and PMIA (Wang et al. 2012) are applicable to the IC model, and SIMPATH (Goyal et al. 2011b) is applicable to the LT model. See a survey (Li et al. 2018) for details.

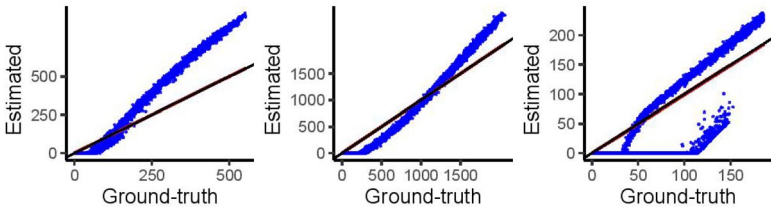
Related to the influence estimation (IE) problem, Yan et al. (2019) proposed a neural network is used to predict the influence of a given seed-node set. While our method relies on a given social network and a specific diffusion model, their method does not assume the observability of the underlying social network or a specific diffusion model. However, they require numerous seed-node sets and ground-truth influence pairs for such generality.

Related to the influence maximization (IM) problem, DISCO (Li et al. 2019) learns a function that estimates the influence of a given seed set and then selects seed nodes that maximize estimated influence based on deep reinforcement learning. GCOMB (Manchanda et al. 2020) leverages a GNN to prune low-quality nodes, followed by a Q-learning module to select high-quality seed nodes under a budget constraint, enabling efficient IM on large-scale graphs. Unlike our method, both approaches are not designed to explicitly estimate the influence of seed sets; instead, they focus on directly selecting seed sets to maximize influence. They are evaluated exclusively under the IC model, with no investigation into their applicability to other diffusion models. Most notably, their learning models are *transductive*, i.e., incapable of estimating influence in social networks unseen during training. In contrast, we aim at designing an *inductive* method, which is capable of estimating the influence of seed nodes in networks whose connections and activation probabilities are completely unseen during training. In addition, our method estimates MC simulation results and thus can be equipped with greedy-based IM algorithms (Kempe et al. 2003; Goyal et al. 2011a; Zhou et al. 2013), whereas the aforementioned methods (Li et al. 2019; Manchanda et al. 2020) directly search for seed nodes.

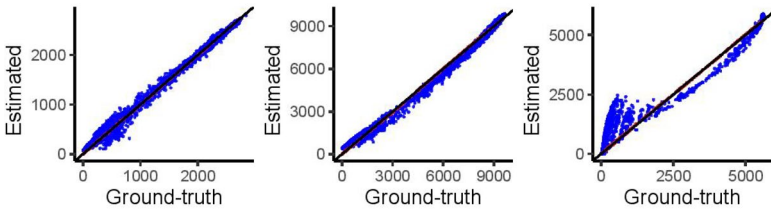
The most relevant to our work is GLIE (Panagopoulos et al. 2023), as it shares the motivation of replacing repeated Monte Carlo simulations with a GNN-based



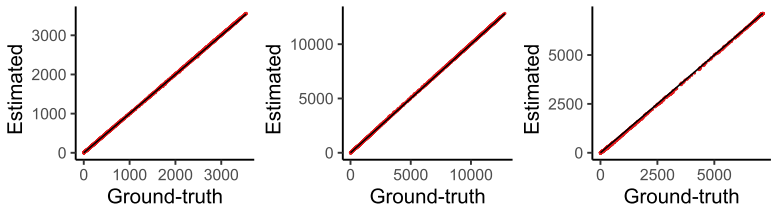
(a) IC model with BT: Extended (left), WannaCry (center), and Celebrity (right)



(b) IC model with JI: Extended (left), WannaCry (center), and Celebrity (right)

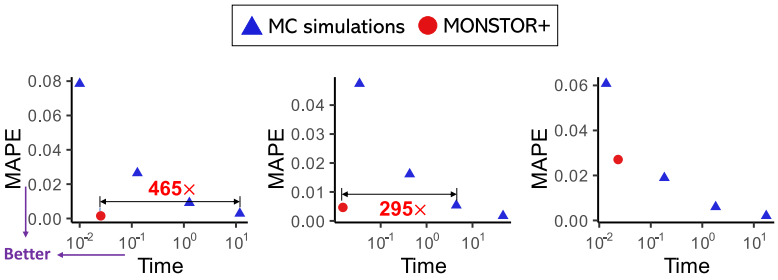


(c) IC model with LP: Extended (left), WannaCry (center), and Celebrity (right)

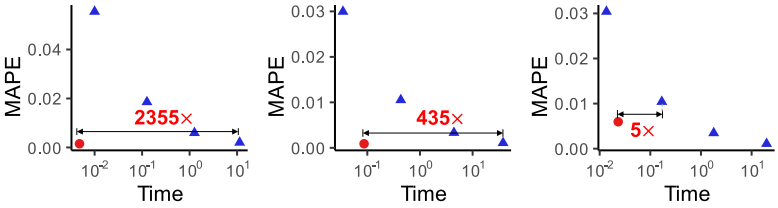


(d) LT model: Extended (left), WannaCry (center), and Celebrity (right)

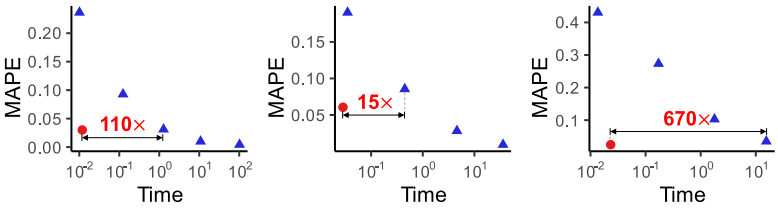
Fig. 8 Scatter plots showing ground-truth influence versus estimates from MONSTOR+ or GLIE on test datasets unseen during training. Red and blue dots represent estimates from MONSTOR+ and GLIE, respectively, while black lines indicate perfect estimation. GLIE is applicable only to the IC model



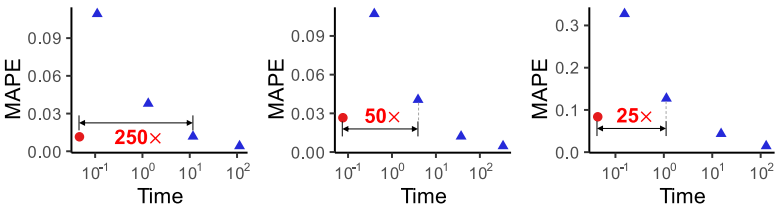
(a) IC model with BT: Extended (left), WannaCry (center), and Celebrity (right)



(b) IC model with JI: Extended (left), WannaCry (center), and Celebrity (right)



(c) IC model with LP: Extended (left), WannaCry (center), and Celebrity (right)



(d) LT model: Extended (left), WannaCry (center), and Celebrity (right)

Fig. 9 Mean absolute percentage error (MAPE) of the influences estimated by MONSTOR+ and repeated MC simulations (with varying numbers of repetitions) in test datasets unseen during training. The further toward the lower-left corner a method is positioned in the plot, the faster and more accurate it is

Table 14 Ablation study. Mean absolute percentage error (MAPE) of the influences estimated by MONSTOR+ and its variants with missing components in test datasets unseen during training

	Auxiliary node feature	Advanced pooling	IC model BT	JI	LP	LT model	Average rank
(a) Extended							
(1)	✓	✓	**0.0015**	0.0015	0.0301	**0.0115**	**1.5**
(2)		✓	0.0023	**0.0010**	**0.0195**	0.0280	**1.5**
(3)	✓		0.0177	0.0092	0.4847	0.0398	3.5
(4)			0.0178	0.0050	0.0325	0.0415	3.5
(b) WannaCry							
	Auxiliary node feature	Advanced pooling	IC model BT	JI	LP	LT model	Average rank
(1)	✓	✓	**0.0047**	0.0009	0.0606	**0.0266**	**1.75**
(2)		✓	**0.0047**	**0.0007**	**0.0314**	0.0458	**1.75**
(3)	✓		0.0096	0.0049	0.1805	0.0455	3.25
(4)			0.0277	0.0028	0.0302	0.0556	3
(c) Celebrity							
	Auxiliary node feature	Advanced pooling	IC model BT	JI	LP	LT model	Average rank
(1)	✓	✓	**0.0271**	0.0060	**0.025**	**0.0837**	**1.5**
(2)		✓	0.0642	**0.0013**	0.1178	0.1684	2.25
(3)	✓		0.0942	0.0045	0.1166	0.5369	3
(4)			0.0624	0.0083	0.2203	0.1845	3.25

Table 15 Ablation study. The influence maximization performance (i.e., the influence of output seeds) of MONSTOR+ and its variants with missing components under the IC model with BT in test datasets unseen during training

	Auxiliary node feature	Advanced pooling	$k = 10$	50	100	Average rank
(a) Extended						
(1)	✓	✓	**481.1 ± 31.1**	**968.5 ± 28.6**	**1223.2 ± 28.8**	**1**
(2)		✓	**481.1 ± 31.1**	**968.5 ± 28.8**	1223.1 ± 29.0	1.33
(3)	✓		479.7 ± 30.3	954.7 ± 29.5	1202.2 ± 29.5	3
(4)			479.7 ± 30.3	943.2 ± 29.5	1195.3 ± 29.5	3.67
(b) WannaCry						
	Auxiliary node feature	Advanced pooling	$k = 10$	50	100	Average rank
(1)	✓	✓	**991.1 ± 68.0**	2123.7 ± 59.1	**2753.3 ± 55.7**	**1.33**
(2)		✓	990.9 ± 68.0	**2123.9 ± 59.3**	2753.1 ± 56.0	1.67
(3)	✓		989.1 ± 59.6	2084.6 ± 55.7	2711.7 ± 54.0	3
(4)			989.0 ± 59.6	2082.0 ± 55.5	2705.2 ± 53.6	4
(c) Celebrity						
	Auxiliary node feature	Advanced pooling	$k = 10$	50	100	Average rank
(1)	✓	✓	**52.8 ± 6.9**	105.1 ± 6.6	155.0 ± 6.6	1.67
(2)		✓	52.6 ± 6.8	**105.2 ± 6.6**	**155.3 ± 6.6**	**1.33**
(3)	✓		47.1 ± 7.3	103.1 ± 6.6	153.2 ± 6.6	3
(4)			45.2 ± 6.6	97.4 ± 6.5	147.3 ± 6.5	4

Table 16 Ablation study. The influence maximization performance (i.e., the influence of output seeds) of MONSTOR+ and its variants with missing components under the IC model with JI in test datasets unseen during training

(a) Extended						
	Auxiliary node feature	Advanced pooling	$k = 10$	50	100	Average rank
(1)	✓	✓	244.4 ± 16.3	**529.4 ± 20.7**	**707.2 ± 22.5**	**1.33**
(2)		✓	**244.5 ± 16.3**	529.3 ± 20.7	**707.2 ± 22.5**	**1.33**
(3)	✓		244.4 ± 16.3	523.4 ± 20.7	695.4 ± 22.8	2.67
(4)			241.8 ± 16.2	515.9 ± 20.9	679.7 ± 22.7	4
(b) WannaCry						
	Auxiliary node feature	Advanced pooling	$k = 10$	50	100	Average rank
(1)	✓	✓	**534.1 ± 24.3**	1239.0 ± 34.3	1648.2 ± 37.2	1.67
(2)		✓	**534.1 ± 24.2**	**1239.1 ± 34.3**	**1648.3 ± 37.3**	**1.33**
(3)	✓		531.4 ± 23.8	1233.6 ± 34.1	1635.5 ± 37.3	3
(4)			528.7 ± 23.7	1232.7 ± 33.9	1629.9 ± 37.0	4
(c) Celebrity						
	Auxiliary node feature	Advanced pooling	$k = 10$	50	100	Average rank
(1)	✓	✓	**43.7 ± 5.8**	**90.4 ± 6.3**	**140.5 ± 6.3**	**1**
(2)		✓	**43.7 ± 5.8**	**90.4 ± 6.3**	140.4 ± 6.3	1.33
(3)	✓		41.9 ± 5.6	89.0 ± 6.2	139.0 ± 6.2	3
(4)			40.0 ± 5.4	87.2 ± 6.1	137.1 ± 6.1	4

estimator with our method. Note that GLIE does not compromise the novelty of our work, as our conference version (Ko et al. 2020) was published earlier. Similar to our methods, GLIE is inductive, allowing a model trained on small (synthetic) graphs to be applied to larger (real-world) graphs at test time. However, while GLIE aims to estimate the total influence of each seed set by being trained on influence–seed set pairs, our methods predict infection probabilities at the node level, using fine-grained (i.e., node-level) supervision that leads to more accurate estimation (see Sects. 4.2 and 4.3 for experimental results). This capability is particularly valuable in applications where understanding the influence on individual nodes is critical, for example, in targeted interventions or personalized marketing strategies. Moreover, while GLIE is limited to the IC model, MONSTOR+ is not restricted to the IC model (see Appendix A.3).

Machine learning approaches, including GNNs (Qiu et al. 2018; Cao et al. 2020), have been used for different but relevant tasks, including predicting the future size of a cascade from its initial stage (Shafiq and Liu 2017; Cao et al. 2020) and estimating the probability that each individual (i.e., node) takes a social action (Qiu et al. 2018). Especially, GNN-based models have been extensively utilized for epidemic forecasting (Panagopoulos et al. 2021; Wang et al. 2022; Xie et al. 2022; Tomy et al. 2022). In this context, nodes of the input graph represent regions, not individuals, and edges indicate geographic adjacency or human mobility. The objective is to predict the number of infection cases that accumulate over time at each region, as opposed to calculating the number of nodes to be infected. Another relevant task, where GNN-

Table 17 The performance (i.e., the influence of output seeds) of IM methods under the G-SIR model. We mark the best performances in **bold**

(a) Extended				
	$k = 5$	10	30	Average rank
Target Influence	3529.3 ± 291.9	3831.0 ± 192.6	4140.2 ± 136.3	-
C-MON+ (Proposed)	**3540.8 ± 291.7**	**3832.2 ± 209.0**	**4192.2 ± 121.6**	**1**
Degree	2674.7 ± 191.7	2731.5 ± 155.1	3721.3 ± 137.6	3.33
Coreness	2312.4 ± 601.8	2641.1 ± 232.5	2862.1 ± 119.6	5
D-SSA	2960.4 ± 473.6	3054.7 ± 411.6	3534.8 ± 211.3	3
SSA	2720.6 ± 744.3	3155.6 ± 319.2	3566.5 ± 158.3	2.67
(b) WannaCry				
	$k = 5$	10	30	Average rank
Target Influence	11485.5 ± 737.7	12032.9 ± 661.2	12590.1 ± 538.4	-
C-MON+ (Proposed)	**11489.2 ± 1028.6**	**11877.0 ± 967.7**	**12659.0 ± 716.4**	**1**
Degree	11048.2 ± 710.5	11294.2 ± 559.3	11814.3 ± 410.1	3.67
Coreness	11120.9 ± 691.1	11442.7 ± 563.1	11739.9 ± 433.6	3.33
D-SSA	10900.0 ± 944.5	11090.7 ± 826.4	11766.3 ± 500.3	4.67
SSA	11151.5 ± 911.0	11376.2 ± 618.3	12103.2 ± 457.4	2.33
(c) Celebrity				
	$k = 5$	10	30	Average rank
Target Influence	6694.1 ± 265.6	6708.5 ± 259.6	6716.5 ± 255.7	-
C-MON+ (Proposed)	**6633.4 ± 315.4**	6713.7 ± 245.3	6738.1 ± 242.8	2.33
Degree	6628.4 ± 309.3	**6730.3 ± 223.8**	**6753.6 ± 209.8**	**1.33**
Coreness	6531.3 ± 435.6	6721.2 ± 237.8	6742.9 ± 222.1	2.33
D-SSA	6399.3 ± 645.1	6607.0 ± 359.8	6720.8 ± 248.9	5
SSA	6483.6 ± 477.1	6634.2 ± 313.1	6721.5 ± 247.3	4

Table 18 The performance (i.e., the influence of output seeds) of IM methods under the considered diffusion models on the Twitter dataset. We mark the best performances in **bold**. In some settings, the target influence could not be computed within a reasonable time and was therefore omitted

(a) IC model				
	$k = 10$	50	100	Average rank
Target Influence	4816.7 ± 886.0	11898.2 ± 821.3	16730.7 ± 755.2	-
C-MON+ (Inductive)	4811.1 ± 842.0	11977.6 ± 823.3	16896.6 ± 753.1	4.33
C-MON+ (Transductive)	**4819.3 ± 887.3**	11989.6 ± 818.8	**16915.5 ± 755.7**	**1.33**
SSA	3772.3 ± 778.9	11896.1 ± 814.3	16840.3 ± 751.7	6
D-SSA	3991.8 ± 837.5	11845.8 ± 815.6	16904.3 ± 754.5	5.33
TIM	4818.7 ± 886.7	**11994.3 ± 821.9**	16912.5 ± 756.7	2
IMM	4819.0 ± 886.7	11986.3 ± 823.6	16906.1 ± 755.7	2.67
IRIE	3058.9 ± 726.2	8975.9 ± 828.2	13580.1 ± 764.7	8
PMIA	2007.3 ± 489.0	7886.3 ± 790.6	11761.7 ± 798.5	9
C-GLIE (Inductive)	4097.0 ± 820.1	9659.2 ± 817.0	14335.2 ± 754.5	6.33
(b) LT model				
	$k = 10$	50	100	Average rank
Target Influence	9779.9 ± 2922.2	23822.5 ± 2893.5	Out of Time	-
C-MON+ (Inductive)	9720.0 ± 2761.0	23981.6 ± 2865.6	32376.3 ± 2459.6	3.67
C-MON+ (Transductive)	9671.0 ± 2748.3	24009.9 ± 2876.5	32445.7 ± 2454.2	2.67
D-SSA	5878.8 ± 2379.9	23895.7 ± 2883.1	32356.9 ± 2525.4	5.33
SSA	7377.8 ± 2659.1	23893.7 ± 2887.5	32295.6 ± 2520.9	5.67
TIM	9835.3 ± 2904.4	**24025.4 ± 2904.1**	**32481.6 ± 2502.2**	**1.33**
IMM	**9839.8 ± 2910.4**	24002.4 ± 2880.8	32421.9 ± 2503.3	2.33
(c) GSIR model				
	$k = 5$	10	30	Average rank
Target Influence	Out of Time	Out of Time	Out of Time	-
C-MON+ (Inductive)	**73376.9 ± 1505.9**	73476.7 + 1476.4	74236.9 ± 1361.4	2.67
C-MON+ (Transductive)	73336.1 ± 1502.3	73518.3 ± 1464.7	**74368.3 + 1341.9**	**2**
Degree	73225.2 ± 1457.3	73330.2 ± 1514.3	73376.8 ± 1508.0	5
Coreness	71142.1 ± 12347.7	71249.3 ± 12293.1	71295.5 ± 12235.8	6
D-SSA	73235.9 ± 1538.8	73658.7 ± 1422.8	74241.6 ± 1318.7	2.67
SSA	73254.7 ± 1527.2	**73683.3 ± 1429.9**	74196.5 ± 1336.3	2.67

based models have been effective, is to detect fake news based on the topology of dynamic diffusion patterns (Han et al. 2020; Sun et al. 2022). There also exist many attempts to predict solutions to various NP-hard problems, and our work was greatly inspired by them. Solutions for the satisfiability, maximal independent set, minimum vertex cover, traveling salesman, and knapsack problems can be predicted by deep learning models (Khalil et al. 2017; Manchanda et al. 2019; Bello et al. 2016; Zoph and Le 2016).

6 Conclusions

In this work, we present MONSTOR+, an inductive machine learning approach for influence estimation (IE) under the independent cascade (IC) and linear threshold (LT) models. We summarize its advantages as follows:

- *Inductive*: Utilizing an inductive machine learning model and features, MONSTOR+ is able to make predictions for seed-node sets and social networks not encountered during training.
- *Fast and accurate*: MONSTOR+ provided near-perfect influence estimation in terms of correlation coefficients (Fig. 8). Especially, compared to Monte Carlo (MC) simulations, MONSTOR+ was 5 to 3000 times faster for similar estimation accuracy (Figs. 6 and 7).
- *Applicable to influence maximization (IM)*: IM algorithms equipped with MONSTOR+, which replaces MC simulations, performed best in 22 out of 27 cases under the IC model and in 7 out of 9 cases under the LT model, among the 10 compared methods (Tables 4-6).

For reproducibility, we make the source code and datasets used in this paper available at https://github.com/SojeongKim00/MONSTOR-plus.

A promising future direction is to enhance the scalability of MONSTOR+, particularly by reducing the cost of its preprocessing step. One approach is to explore more scalable feature alternatives to local cycles, whose computation incurs cubic time complexity.

Additional experimental results

Influence estimation accuracy

Figure 8 presents scatter plots comparing ground-truth and estimated influences on test datasets unseen during training under various diffusion models (refer to Sect. 4.2 for detailed settings). Figure 9 shows the mean absolute percentage error (MAPE) of the influences estimated by MONSTOR+ and repeated MC simulations, with varying numbers of repetitions, (refer to Sect. 4.2.2 for detailed settings).

Ablation study

Table 14 shows the mean absolute percentage error (MAPE) of the influences estimated by MONSTOR+ and its variants with missing components in test datasets unseen during training. Tables 15 and 16 present the influence maximization performances (i.e., the influence of output seeds) of MONSTOR+ and its variants with missing components under the IC model with BT and JI, respectively, in test datasets unseen during training. Refer to Sect. 4.4 for detailed settings.

Extension to another diffusion model

In this section, to show the generality of MONSTOR+, we extend it to another diffusion model: the general Markov chain susceptible-infected-recovered (G-SIR) (Yi et al. 2022) model.

The G-SIR model generalizes the classic SIR diffusion process by allowing edge-specific propagation probabilities. In the original SIR model, the propagation probability is uniform across all edges, whereas in the G-SIR model, each edge can have a distinct propagation probability. Under the G-SIR model, every active node simultaneously attempts to infect each of its out-neighbors with the corresponding activation probabilities and recovers independently with a recovery rate r. Once the activated nodes recover, they remain inactive for the rest of the diffusion process. As long as each activated node remains active, it can continue to infect other non-recovered nodes. The process continues until no further infections occur.

Support for the G-SIR model (or any diffusion model) is achieved by generating training and validation data via Monte Carlo simulations of the corresponding diffusion process. Once the data are obtained, we train and stack our GNN model following the pipeline described in Sect. 3.1.

We empirically evaluate the effectiveness of MONSTOR+ under the G-SIR model through comparisons with baseline methods. MONSTOR+ is combined with CELF as an influence estimator (we refer to the combination as C-MON+); however, its theoretical guarantees under the IC and LT models do not hold for the G-SIR model due to its non-submodularity. Considering the applicability to the G-SIR model, we use the following IM algorithms as competitors under G-SIR model: (1) CELF (Goyal et al. 2011a) among simulation-based algorithm as target influence; (2) SSA (Nguyen et al. 2016) and D-SSA (Nguyen et al. 2016) among non-simulation based methods, and (3) degree and coreness as heuristic methods. For the baselines, we follow the settings in the main paper.

In our experiments, the activation probability in the G-SIR model follows the weighted cascade model (Kempe et al. 2003), where the weight of each edge from node u to v is set to the inverse of the in-degree of v. The recovery rate is chosen uniformly at random from the range 0.28 to 0.35. In the G-SIR model, infected nodes continue to spread the infection until they recover, resulting in relatively high influence even with a small number of seeds. To account for this, we use fewer seed nodes than in other models and evaluate IM performance with 5, 10, and 30 seeds.

As shown in Table 17, our method (C-MON+) achieved the best performance on all datasets except Celebrity. Although it was not the top performer on every individual dataset, it demonstrated the best overall average performance across all datasets.

Experiments on a large-scale dataset

In this section, we present additional influence maximization (IM) experiments on a large-scale dataset, Twitter (Leskovec and Mcauley 2012), available from SNAP (Leskovec and Krevl 2014). The dataset consists of 81, 306 nodes and 1, 768, 149 edges.

Since the dataset does not provide (approximate) activation probabilities, we use the weighted cascade model (Kempe et al. 2003), in which the activation probability

of each edge is set to the reciprocal of the in-degree of the target node, in addition to the LT and G-SIR models. For our method, MONSTOR+, we consider two settings: (1) the transductive setting, where the model is trained and tested on the Twitter dataset, and (2) the inductive setting, where the model is trained on smaller datasets (Extended, Celebrity, and WannaCry) and tested on the larger Twitter dataset. The detailed settings, including baseline parameters, are kept the same as those used in Sect. 4.1.

As shown in Table 18, C-MON+ (i.e., MONSTOR+ applied to CELF for influence maximization) achieves the best performance in two out of the three diffusion models under the transductive setting. In the inductive setting, although C-MON+ is outperformed by several sketch-based methods, it significantly outperforms the other learning-based method, C-GLIE, under the IC model, where C-GLIE is applicable.

Author contributions J. Ko and S. Kim wrote the draft of the paper, designed experiments, and prepared figures and tables. J. Ko, S. Kim, K. Lee, S. Kang, and D. Hwang performed experiments and analysis. N. Park and K. Shin revised the manuscript. All authors designed the methodology and reviewed the manuscript.

Funding Open Access funding enabled and organized by KAIST. Open Access funding enabled and organized by KAIST. This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00438638, EntireDB2AI: Foundations and Software for Comprehensive Deep Representation Learning and Prediction on Entire Relational Databases, 50%) (No. RS-2022-II220157, Robust, Fair, Extensible Data-Centric Continual Learning, 40%) (No. RS-2019-II190075, Artificial Intelligence Graduate School Program (KAIST), 10%).

Data availability https://github.com/SojeongKim00/MONSTOR-plus

Declarations

Conflict of interest The authors declare no Conflict of interest.

References

Bello I, Pham H, Le QV, et al. (2016) Neural combinatorial optimization with reinforcement learning. *arXiv preprint* arXiv:1611.09940https://doi.org/10.48550/arXiv.1611.09940

Cao Q, Shen H, Gao J, et al. (2020) Popularity prediction on social platforms with coupled graph neural networks. In: *Proceedings of the 13th International Conference on Web Search and Data Mining*. ACM, pages 70–78, https://doi.org/10.1145/3336191.3371834

Chakrabarti D, Zhan Y, Faloutsos C (2004) R-mat: A recursive model for graph mining. In: *Proceedings of the 2004 SIAM International Conference on Data Mining (SDM)*, pages 442–446, https://doi.org/10.1137/1.9781611972740.43

Chen W, Wang C, Wang Y (2010) 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 (KDD)*. ACM, pages 1029–1038, https://doi.org/10.1145/1835804.1835934

Corso G, Cavalleri L, Beaini D et al (2020) Principal neighbourhood aggregation for graph nets. Adv Neural Inform Process Syst NIPS 33:13260–13271

Domingos P, Richardson M (2001) Mining the network value of customers. In: *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, pages 57–66, https://doi.org/10.1145/502512.502525

Dwivedi VP, Luu AT, Laurent T, et al. (2021) Graph neural networks with learnable structural and positional representations. In: *International Conference on Learning Representations (ICLR)*, https://doi.org/10.48550/arXiv.2110.07875

Easley D, Kleinberg J et al (2010) Networks, crowds, and markets: reasoning about a highly connected world, vol 1. Cambridge University Press, Cambridge

Goyal A, Lu W, Lakshmanan LV (2011a) Celf++: Optimizing the greedy algorithm for influence maximization in social networks. In: *Proceedings of the 20th international conference companion on World wide web (WWW)*. ACM, pages 47–48, https://doi.org/10.1145/1963192.1963217

Goyal A, Lu W, Lakshmanan LV (2011b) Simpath: An efficient algorithm for influence maximization under the linear threshold model. In: *2011 IEEE 11th International Conference on Data Mining (ICDM)*. IEEE, pages 211–220, https://doi.org/10.1109/ICDM.2011.132

Han Y, Karunasekera S, Leckie C (2020) Graph neural networks with continual learning for fake news detection from social media. *arXiv preprint* arXiv:2007.03316https://doi.org/10.48550/arXiv.2007.03316

He X, Song G, Chen W, et al. (2012) Influence blocking maximization in social networks under the competitive linear threshold model. In: *Proceedings of the 2012 SIAM International Conference on Data Mining (SDM)*, pages 463–474, https://doi.org/10.1137/1.9781611972825.40

Jung K, Heo W, Chen W (2012) Irie: Scalable and robust influence maximization in social networks. In: *2012 IEEE 12th International Conference on Data Mining (ICDM)*. IEEE, pages 918–923, https://doi.org/10.1109/ICDM.2012.79

Kempe D, Kleinberg J, Tardos E (2003) Maximizing the spread of influence through a social network. In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, pages 137–146, https://doi.org/10.1145/956750.956769

Khalil E, Dai H, Zhang Y, et al. (2017) Learning combinatorial optimization algorithms over graphs. Adv Neural Inform Process Syst 30

Ko J, Lee K, Shin K, et al. (2020) Monstor: an inductive approach for estimating and maximizing influence over unseen networks. In: *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, pages 204–211, https://doi.org/10.48550/arXiv.2001.08853

Leskovec J, Krevl A (2014) SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data

Leskovec J, Mcauley J (2012) Learning to discover social circles in ego networks. Adv Neural Inform Process Syst 25

Li H, Xu M, Bhowmick SS, et al. (2019) Disco: Influence maximization meets network embedding and deep learning. *arXiv preprint*arXiv:1906.07378 https://doi.org/10.48550/arXiv.1906.07378

Li Y, Zhang D, Tan KL (2015) Real-time targeted influence maximization for online advertisements. In: *Proceedings of the VLDB Endowment* p 1070-1081. https://doi.org/10.14778/2794367.2794376

Li Y, Fan J, Wang Y et al (2018) Influence maximization on social graphs: a survey. IEEE Trans Knowl Data Eng (TKDE) 30:1852–1872. https://doi.org/10.1109/TKDE.2018.2807843

Liu J, Chen Y, Li D, et al. (2019) Predicting influence probabilities using graph convolutional networks. In: 2019 IEEE International Conference on Big Data (Big Data), IEEE, pages 860–869

Manchanda S, Mittal A, Dhawan A, et al. (2019) Learning heuristics over large graphs via deep reinforcement learning. *arXiv preprint*arXiv:1903.03332 https://doi.org/10.48550/arXiv.1903.03332

Manchanda S, Mittal A, Dhawan A et al (2020) Gcomb: learning budget-constrained combinatorial algorithms over billion-sized graphs. Adv Neural Inf Process Syst 33:20000–20011

Nguyen HT, Thai MT, Dinh TN (2016) Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In: *Proceedings of the 2016 International Conference on Management of Data (SIGMOD)*. ACM, pages 695–710, https://doi.org/10.1145/2882903.2915207

Ou HC, Chou CK, Chen MS (2016) Influence maximization for complementary goods: Why parties fail to cooperate? In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM)*. ACM, pages 1713–1722, https://doi.org/10.1145/2983323.2983741

Panagopoulos G, Nikolentzos G, Vazirgiannis M (2021) Transfer graph neural networks for pandemic forecasting. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 4838–4845, https://doi.org/10.1609/aaai.v35i6.16616

Panagopoulos G, Tziortziotis N, Vazirgiannis M, et al. (2023) Maximizing influence with graph neural networks. In: *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 237–244

Qiu J, Tang J, Ma H, et al. (2018) Deepinf: Social influence prediction with deep learning. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, pages 2110–2119, https://doi.org/10.1145/3219819.3220077

Sabottke C, Suciu O, Dumitras T (2015) Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In: *Proceedings of the 24th USENIX Conference on Security Symposium (USENIX Security)*. USENIX Association, pages 1041–1056, https://doi.org/10.5555/2831143.2831209

Sankar CP, Kumar KS (2016) Learning from bees: an approach for influence maximization on viral campaigns. PLoS ONE 11(12):e0168125. https://doi.org/10.1371/journal.pone.0168125

Shafiq Z, Liu A (2017) Cascade size prediction in online social networks. In: *2017 IFIP Networking Conference and Workshops (IFIP Networking)*, pages 1–9, https://doi.org/10.23919/IFIPNetworking.2017.8264864

Sun M, Zhang X, Zheng J, et al. (2022) Ddgcn: Dual dynamic graph convolutional networks for rumor detection on social media. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 4611–4619, https://doi.org/10.1609/aaai.v36i4.20385

Tang Y, Xiao X, Shi Y (2014) Influence maximization: Near-optimal time complexity meets practical efficiency. In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD)*. ACM, pages 75–86, https://doi.org/10.1145/2588555.2593670

Tang Y, Shi Y, Xiao X (2015) Influence maximization in near-linear time: A martingale approach. In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD)*. ACM, pages 1539–1554, https://doi.org/10.1145/2723372.2723734

Tomy A, Razzanelli M, Di Lauro F et al (2022) Estimating the state of epidemics spreading with graph neural networks. Nonlinear Dyn. https://doi.org/10.1007/s11071-021-07160-1

Wang C, Chen W, Wang Y (2012) Scalable influence maximization for independent cascade model in large-scale social networks. Data Min Knowl Discov (DAMI). https://doi.org/10.1007/s10618-012-0262-1

Wang L, Adiga A, Chen J, et al. (2022) Causalgnn: Causal-based graph neural networks for spatio-temporal epidemic forecasting. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 12,191–12,199, https://doi.org/10.1609/aaai.v36i11.21479

Wilder B, Vorobeychik Y (2018) Controlling elections through social influence. In: *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, https://doi.org/10.48550/arXiv.1711.08615

Xie F, Zhang Z, Li L, et al. (2022) Epignn: Exploring spatial transmission with graph neural network for regional epidemic forecasting. In: *Joint European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*. Springer, pages 469–485, https://doi.org/10.1007/978-3-031-26422-1_29

Xu K, Zhang M, Li J, et al. (2020) How neural networks extrapolate: From feedforward to graph neural networks. In: *International Conference on Learning Representations (ICLR)*, https://doi.org/10.48550/arXiv.2009.11848

Yadav A, Chan H, Xin Jiang A, et al. (2016) Using social networks to aid homeless shelters: Dynamic influence maximization under uncertainty. In: *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, https://doi.org/10.5555/2936924.2937034

Yan B, Song K, Liu J, et al. (2019) On the maximization of influence over an unknown social network. In: *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 2279–2281, https://doi.org/10.5555/3306127.3332084

Yi Y, Shan L, Paré PE et al (2022) Edge deletion algorithms for minimizing spread in sir epidemic models. SIAM J Control Optim 60(2):S246–S273

You J, Gomes-Selman JM, Ying R, et al. (2021) Identity-aware graph neural networks. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, https://doi.org/10.48550/arXiv.2101.10320

Zhou C, Zhang P, Guo J, et al. (2013) Ublf: An upper bound based approach to discover influential nodes in social networks. In: *2013 IEEE 13th International Conference on Data Mining (ICDM)*. IEEE, pages 907–916, https://doi.org/10.1109/ICDM.2013.55

Zoph B, Le QV (2016) Neural architecture search with reinforcement learning. *arXiv preprint* arXiv:1611.01578https://doi.org/10.48550/arXiv.1611.01578

Authors and Affiliations

Jihoon Ko[1] · Sojeong Kim[1] · Kyuhan Lee[1] · Shinhwan Kang[1] · Dongyeong Hwang[1] · Kijung Shin[1] · Noseong Park[2]

✉ Kijung Shin
kijungs@kaist.ac.kr

✉ Noseong Park
noseong@kaist.ac.kr

Jihoon Ko
jihoonko@kaist.ac.kr

Sojeong Kim
kimsojeong@kaist.ac.kr

Kyuhan Lee
kyuhan.lee@kaist.ac.kr

Shinhwan Kang
shinhwan.kang@kaist.ac.kr

Dongyeong Hwang
dy.hwang@kaist.ac.kr

[1] Kim Jaechul Graduate School of AI, KAIST, Seoul, South Korea

[2] School of Computing, KAIST, Daejeon, South Korea