Data Mining and Knowledge Discovery (2025) 39:71
https://doi.org/10.1007/s10618-025-01133-3

Topic-aware influence maximization with deep reinforcement learning and graph attention networks

Taha Halal[1] · Bogdan Cautis[2] · Benoît Groz[1] · Ruize Gao[3]

Received: 21 April 2025 / Accepted: 9 July 2025

Abstract
Influence maximization is a fundamental problem in network analysis, focusing on identifying a subset of nodes in a social network to maximize the spread of influence. In this paper, we present an approach for tackling the Influence Maximization (IM) problem, integrating Deep Reinforcement Learning (DRL) techniques with attentive Graph Neural Networks (GATs). Our study builds upon a prior algorithm (S2V-DQN-IM) and progressively refines it towards IM-GNN, ultimately achieving competitive performance against state-of-the-art methods on classic IM. Through experiments on benchmark datasets, we empirically validate the effectiveness of graph attention mechanisms and positional encoding, using the graph magnetic Laplacian, to reach state-of-the-art performance in terms of influence spread. Building on this success, we extend our IM-GNN framework to incorporate *topic-awareness* in TIM-GNN, recognizing the inherent topical nature of real-world diffusions. By harnessing probabilistic techniques, we construct topic-aware social graphs using real cascades and assess the effectivenesss of TIM-GNN on them. Our extensive experimental results validate the utility of our topic-aware approach, demonstrating significant advances over existing topic-aware IM methods. Finally, in order to improve upon performance (latency) at query time, we develop a variant of TIM-GNN, called TIM-GNN^x, by using *cross*-attention mechanisms. We show it maintains comparable overall spread performance as its predecessor, while achieving a 10x-20x speed-up.

Keywords Information diffusion · Graph neural networks · Deep RL · Topic-aware

Taha Halal and Bogdan Cautis have contributed equally to this work.

Extended author information available on the last page of the article

Published online: 19 August 2025

1 Introduction

Social media has redefined the ways in which information spreads. By word-of-mouth mechanisms, viral spread can happen rapidly and at unprecedented scale. Whether for advertising or political / public-awareness campaigns, understanding and taking advantage of this highly effective medium for information diffusion is paramount. Unsurprisingly, the key for a successful diffusion campaign is to know where to start, i.e., to choose the best sources (*spread seeds*) from which to initiate the dissemination of information. Under the generic name of Influence Maximization (in short, IM), we have seen in recent years many advances, at the intersection of combinatorial optimization on graphs and network analysis (Kempe et al. 2003).

Despite diverse formulations, most IM studies share a common focus: optimally select k seed nodes in a *diffusion graph*– a directed graph whose edges are labelled by a diffusion probability, while optimality usually boils down to maximizing the expected spread under a specific diffusion model, such as Independent Cascades.

While conceptually simple and extensively studied, IM remains notoriously difficult, with few solutions being truly applicable in real-life scenarios. IM is in general NP-hard Kempe et al. (2003), and most studies exploit the monotonicity and submodularity of the spread objective, which guarantee the greedy selection algorithm to achieve at least 63% of the optimal spread. However, the greedy approach itself remains computationally expensive, as one key step therein– the estimation of the *marginal spread gain* for a candidate seed w.r.t. to the current partial solution– is #P-hard and requires in practice expensive sampling steps (Monte Carlo, RR sets, etc) Leskovec et al. (2007); Tang et al. (2015).

As a possible direction for IM solutions that can be applied *at scale* to answer in *real-time* IM queries, we consider an approach that aims to replace the expensive estimation of marginal gain by a prediction thereof. In short, we want to pre-train a model that can predict *at query time* a node's quality in a given *context*, namely for a given diffusion graph, IM query, and partial seed set. With likely but hopefully acceptable loss in effectiveness, the trade-off would be to gain significantly in efficiency.

In essence, we aim to train a predictive model $\mathcal{M}$ for marginal gain, by DRL. However, training the model is computationally expensive, especially on large graphs. We argue that this upfront cost can be justified only if it is amortized over multiple queries. Our approach is thus akin to some of the existing IM solutions, which precompute and index key information to speed up computation at query time (but the key information in our framework is a model trained by DRL). More precisely, we adopt a 3-staged approach.

1. in a *training stage*, we build the predictive model $\mathcal{M}$ using well-chosen and manageable samples of the diffusion graph $\mathcal{G}$; this stage may be slow.
2. in a *pre-query / embedding stage*, we build the representation of the entire input graph $\mathcal{G}$ (on which IM queries are to be computed), to be given as input to $\mathcal{M}$; this stage may be quite fast, yet not real-time.
3. in a *query stage*, IM queries Q would be answered, using $\mathcal{G}$'s embedding and $\mathcal{M}$; this must be fast for each query.

In light of this, two important questions need to be clarified:

Q1 As our primary motivation stems from the scalability and efficiency limitations of classic IM solutions, *is the idea of training such a model $\mathcal{M}$ justified for real-time IM querying*?
Q2 As social graphs are highly dynamic, *how robust can the model $\mathcal{M}$ be, with respect to diffusion graph changes*?

We believe that [Q1] can be answered positively only if we have a large space of potential IM queries. Vanilla IM queries (i.e., under a traditional IM formulation, without other context dimensions such as location, topic, or time) simply amount to a budget value k, but most diffusions in social media are *topic-aware*. Indeed, in real-world applications, diffusions convey messages with various topical distributions, making it necessary to handle a much larger space of potential IM queries. Therefore, our thesis is that answering Q1 positively– for efficient / real-time query response– within the DRL framework *hinges* on topic-aware diffusion models.

Regarding [Q2], $\mathcal{M}$ must be *robust* to graph evolution: if trained on a snapshot of the diffusion graph at time t, it must be able to make predictions on future snapshots of that graph, which most likely remain structurally similar. We stress that we focus on *robustness* instead of *generalizability*– i.e., the ability of $\mathcal{M}$ to make predictions on other graphs, unseen at training and maybe structurally different– as we believe the former is more important for practical scenarios. E.g., there is little practical interest in training on a Twitter diffusion graph and testing on a Meta Threads one.

Our contributions. We revisit in this paper the generic framework S2V-DQN of Dai et al. (2017), which designs heuristic algorithms for graph-based combinatorial optimization (CO) problems using DRL.

IM-GNN. First, as a proof-of-concept, we build upon S2V-DQN and progressively refine the initial and direct adaptation of it to IM (generically called **S2V-DQN-IM**) towards our first approach called **IM-GNN**, which shows competitive performance w.r.t. the existing learning-based methods for *vanilla IM*. In particular, we integrate in *IM-GNN* attentive GNNs and positional encoding with the novel graph magnetic Laplacian (He et al. 2022). Importantly, for our comparison of IM-GNN with SOTA baseline methods, we use public, real-world datasets and diffusion graphs built from cascades, i.e., with *data-based* diffusion probabilities. In contrast, the graphs used in the evaluation of existing methods are mostly artificial and topology-bound, with uniform, trivalency, or degree-based edge probabilities.

TIM-GNN. Building on the confirmation of IM-GNN's effectiveness, we extend our framework to incorporate *topic-awareness*, leading to our method **TIM-GNN**. For a realistic evaluation, from public data, we extract topic-aware diffusion graphs from information cascades, using the survival factorization framework of Barbieri et al. (2017). We assess the performance of TIM-GNN on (1) effectiveness (spread), (2) efficiency (query latency), and (3) robustness to changes in the diffusion graph. The existing baseline methods we use for comparison are either *topic-agnostic learning-based* or *non-learning based topic-aware*. Our experimental results show that TIM-GNN can meet the stringent requirements of real-world applications, being superior to the state-of-the-art, albeit relatively slow at query time.

TIM-GNN.^{x} Finally, to improve latency, we incorporate *cross-attention mechanisms*. We refine TIM-GNN to accurately predict a high-quality ranking of seed nodes for any query (item and budget combination), by using a pre-computed Q-matrix derived from well-chosen representative *base items*. The resulting approach (**TIM-GNN**^{x}) allows for real-time yet effective assessment of influence spread, significantly reducing the need for extensive graph message passing operations during each assessment. As a result, we mostly preserve the robust performance of our first-cut topic-aware algorithm TIM-GNN, while achieving a $10\times - 20\times$ query time speed-up.

2 Related work

Influence maximization (IM) was first formalized in Kempe et al. (2003), and shown to be NP-hard, for diffusion models such as Linear Threshold (LT) or Independant Cascades (IC). Exploiting the monotonicity and submodularity of the influence spread function, they show that a simple greedy strategy is within $(1 - 1/e)$ of the optimal. Since diffusion models are stochastic, Monte Carlo simulations are needed to estimate the spread (marginal gain) of a candidate seed. For traditional (vanilla) IM formulation, the majority of the subsequent studies focused on efficiency. CELF Leskovec et al. (2007) exploits submodularity, leading to far fewer diffusion simulations in practice. IMM Tang et al. (2015) introduces estimation techniques based on martingales, which enable to maintain the worst-case guarantees of the greedy approach, while increasing the efficiency, based on reverse influence sampling (RIS) Borgs et al. (2012). It *reversely* samples a group of influence paths– reverse reachable (RR) sets– and then looks for the seed set that overlaps the most with these sets. While these and numerous other good heuristics were proposed in recent years, their applicability in real-world IM settings at scale remains problematic, due to the intrinsic hardness of the combinatorial problem combined with the stochasticity of models. See Ohsaka (2020); Arora et al. (2017) for an in-depth comparison of traditional IM methods.

Topic-aware IM. Vanilla IM models do not capture variations in diffusion patterns, essentially treating diffusion items as a black box (ignoring content), yet topical features have been shown to play a major role in how information may spread virally (Du et al. 2013). Extending the vanilla IC model, Topic-aware Independent Cascades (TIC) diffusion models were proposed initially in Barbieri et al. (2013) and further refined in Barbieri et al. (2017). In short, TIC assumes that each edge is labelled by a probability diffusion *vector* of dimension d (number of topics), with each component denoting the activation probability from the source node to the target one under a specific topic. While capturing more closely real-world diffusions than vanilla IM models, TIC has the downside of bringing an additional level of complexity to an already challenging problem. Algorithms for online topic-aware IM have been proposed, based on TIC, following generally a strategy of pre-processing and indexing topic-agnostic IM results, such as INFLEX (from INFluence indEX) Aslay et al. (2014), as well as Chen et al. (2015a, 2015b) (see Li et al. (2018) for a survey on IM involving other relevant dimensions of social networks, such as topics). We

contribute to this line of research by proposing a DRL-based approach, predicting marginal spread gain under TIC, based jointly on the diffusion medium and the topical profile of the item to spread.

S2V-DQN. Good heuristics for CO on graphs often require tremendous specialized knowledge and trial-and-error. Dai et al. (2017) presents a new perspective on CO problems in general. They use a powerful end-to-end trainable approach called S2V-DQN, merging the strengths of Deep Q-Network (DQN) in reinforcement learning with Structure2Vec's (S2V) capability in graph embedding. It acts as a meta-algorithm, capable of automating the design of heuristics and incrementally constructing a solution for hard CO problems on graphs. *In our paper, we adopt and adapt this framework to the problem of topic-aware IM.*

Learning-based methods for IM. We've seen in recent years an inflation of learning-based methods for IM, alas, with many of them suffering in our view from similar conceptual, methodological, or scientific rigour shortcomings. A recent survey of such approaches can be found in Li et al. (2023). We mostly focus our discussion here on the deep reinforcement learning (DRL) research that builds upon the meta-algorithm S2V-DQN, adapting it for IM problems. PIANO (deeP reInforcement leArning-based iNfluence maximizatiOn) is a DQN-based IM algorithm proposed by Li et al. (2022), and the first study to use the S2V-DQN framework for IM. It trains the DQN model on small, topologically coherent samples of the diffusion graph, before evaluating it on the full graph. With a focus on generalizability, PIANO also proposes to maintain a collection of pre-computed models to avoid retraining. Li et al. (2022) claims superior efficiency and comparable result quality with state-of-the-art classical IM methods. However, we found serious issues while trying to reproduce the paper's results on benchmark IM datasets. Our basic algorithmic design (S2V-DQN-IM) for vanilla IM is quite similar to PIANO conceptually, which allows us to follow the performance gains brought to it by our progressively refined methods IM-GNN and TIM-GNN. DIEM (Deep Influence Evaluation Model) Tian et al. (2020) is the only study to adapt the S2V-DQN framework for topic-aware IM. However, it advances only marginally the general understanding of the problem space and potential solutions, due to experimental limitations and many unspecified details. For example, DIEM is evaluated on undisclosed Twitter data, under an ad-hoc topic-aware model, with artificial topology-based diffusion probabilities.[1] Overall, PIANO and DIEM have similar limitations. In particular, they both evaluate on artificial diffusion graphs, which limits their relevance for practical IM scenarios. Instead, we focus on cascade-based diffusion graphs, built from public cascading datasets. Moreover, many unclear aspects remain on how these studies extend and improve upon the S2V-DQN framework. In fact, PIANO does not discuss this at all; incidentally, it does not cite Dai et al. (2017), although it adopts its conceptual framework.

ToupleGDD (**T**hree C**oupl**ed **G**raph Neural Networks with **D**ouble **D**eep Q-networks) Chen et al. (2023) is an S2V-DQN-like framework for vanilla IM that couples three GNNs enhanced with an attention mechanism for network embedding. It has a particular focus on generalizability, and therefore trains on small, randomly generated

[1] We could not obtain the code / data from the authors, their results are not reproducible, and a comprehensive evaluation using DIEM as a baseline method proved impossible.

graphs and with a small seed set budget. This allows it to be evaluated on a multitude of diffusion graphs and for various budgets. GCOMB Manchanda et al. (2020) is an approach for vanilla IM, which, unlike S2V-DQN, does not learn end-to-end. It starts from the observation that pruning the search space is as important as the prediction of the solution. GCOMB works by first using a graph convolutional network (GCN) to identify promising nodes (as seed set candidates); the GCN learns embeddings of the promising nodes in a supervised manner. Then, it uses a Q-learning component to predict the solution set from the promising nodes. The mixture of supervised and RL makes GCOMB on one hand lightweight (fewer parameters), but on the other hand highly dependent on the quality of the supervised stage. The DeepIM approach of Ling et al. (2023) tackles vanilla IM on a fixed graph using an autoencoder to compress seed sets, training the network to optimize the spread prediction from the reconstructed seed set. The optimal seed set is then inferred through gradient descent in the low-dimension space. By design, one particular focus of DeepIM is generalizability under various diffusion models, which may impact its predictive performance in certain settings. While alternative diffusion models are beyond our scope, we see this additional level of robustness– across diffusion models– as an interesting direction for future research. *ToupleGDD, GComb, and DeepIM are the most recent and, in our view, the best representatives of the state-of-the-art on learning-based IM; we will employ them as baseline methods.*

Other related work. RL4IM Chen et al. (2021) is an RL approach for *contingency-aware* IM, where the seeded nodes accept their role with a certain probability. They focus on *reward shaping* since, due to the contingency assumption, it is no longer possible to use the marginal spread of a newly selected node as the immediate reward at each step. When contingency is relaxed, RL4IM becomes the basic S2V-DQN we compare with. IMInfector Panagopoulos et al. (2022) learns topic-agnostic diffusion probabilities from cascades, making it a potential complementary (learning-based) way for getting the probabilities for our framework. The DeepIS approach (for **D**eep **I**nfluence **S**usceptibility) of Xia et al. (2021) predicts the *susceptibility* of target nodes to be influenced by a seed set. They approach the problem as a node regression task in the GNN framework.

Positioning w.r.t state-of-the-art. While there is an extensive and diverse work on new DRL methods for graph CO problems (e.g., for vanilla IM), in particular building upon the simple, generic S2V-DQN framework, our work represents the first attempt to leverage DRL for topic-aware IM. We argue that a DRL-based approach within this framework is only justifiable if there is a large space of potential IM queries (hence topic-aware), which can be answered in real-time. The aforementioned DRL-based solutions for vanilla IM cannot be directly applied nor easily adapted to address the challenges of topic-aware IM. In essence, TIM-GNN^x is the first approach addressing the three critical challenges for practical applicability of learning-based IM solutions: predictive *accuracy* over a broad (topic-aware) query space, *low latency* over large diffusion graphs, and *robustness* to graph changes. As illustrated in our experiments, existing methods fall short in at least one of these aspects. Importantly, in these experiments we incorporate realistic diffusion graphs, either topic-agnostic or topic-aware, in contrast with the commonly used uniform or

trivalency probability models of most prior works (including all DRL-based methods for vanilla IM).

3 Topic-aware influence maximization

Diffusion networks and IM. A social (diffusion) network is commonly defined as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{P})$, where $\mathcal{V}$ denotes the nodes (users), $\mathcal{E}$ is the set of edges between them, and $\mathcal{P}$ is a probability function on $\mathcal{E}$, such that information propagates along an edge (u, v) according to the edge probability $p_{u,v}$. Information or influence spread is therefore the result of a stochastic process, following a *diffusion model*. Then, *Influence Maximisation (IM)* can be defined generically as follows: select a set of seed nodes $S \subseteq \mathcal{V}$, of size at most k, such that the expected spread of influence starting from S (or the expected number of activated nodes) is maximized.

Independent Cascades (IC). IC is the most well-known diffusion model. In IC, each node can be in one of two states: active or inactive. At the initial time step, only the nodes in the seed set S are active. At each subsequent time step, all nodes that transitioned from inactive to active at the previous time step will independently make a unique attempt to activate each of their inactive neighbors. A neighbor will be activated with probability equal to the weight of the edge between the two nodes. The propagation ends when no nodes can activate any of their neighbors. The *spread* $\sigma(S)$ is the number of nodes that are activated at the end of the propagation.

Topic-aware IM. As an extension to IC, we consider the topic-aware model TIC, initially proposed in Barbieri et al. (2013), which takes into consideration the topical description of the information being diffused. TIC assumes that each edge in the graph may spread information pertaining to a certain number d of topics: namely, $(u, v) \in \mathcal{E}$ is associated with a topic spreading-weight vector $\mathrm{p}_{u,v} = \left(p^1_{u,v}, p^2_{u,v}, \ldots, p^d_{u,v}\right)$, $0 \leq \mathrm{p}_{u,v} \leq 1$, where $p^z_{u,v}$ is the weight associated to topic z, denoting the probability for topic z that user u activates user v.

Given an item, i.e., a topic distribution vector $\overrightarrow{\gamma}$ as the information that is diffused, $\overrightarrow{\gamma} = \left(\gamma^1, \gamma^2, \ldots, \gamma^d\right)$ s.t. $\sum_{1 \leq i \leq d} \gamma^i = 1$, for each edge (u, v), the propagation probability along that edge w.r.t. $\overrightarrow{\gamma}$ is:

$$p_{u,v}(\overrightarrow{\gamma}) = \langle \mathrm{p}_{u,v}, \overrightarrow{\gamma} \rangle = \mathrm{p}^{\top}_{u,v} \overrightarrow{\gamma} \tag{1}$$

It is this propagation probability that will be used for edge (u, v), as described before, in a *TIC diffusion process*. When, for a given item $\overrightarrow{\gamma}$ the Eq. (1) is applied to all the edges of $\mathcal{G}$, we say the item is **projected** on $\mathcal{G}$, leading to a topic-agnostic diffusion graph (having a single diffusion probability per edge). Consequently, in topic-aware IM, the objective is to find the best seed set given an item-budget query $\mathcal{Q} = (\overrightarrow{\gamma}, k)$, and we can define $\sigma(S|\overrightarrow{\gamma})$ similarly.

Greedy algorithm and diffusion simulations. As the objective function in IM is monotone submodular, the general approach for finding an approximate solution is based on the greedy algorithm. It selects at each step a new seed node that yields the largest marginal gain on expected spread w.r.t. the current partial solution. How-

ever, computing the expected spread is #P-hard for IC (and clearly for TIC as well), so most traditional IM research works focus on approximation methods. One such method is to simulate r random cascades from a given seed node and average the number of influenced nodes to approximate the marginal spread.

We can formally define topic-aware IM as follows:

Problem 1 *(Topic-aware IM) Given the $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{P})$ topic-aware network and a query $\mathcal{Q} = (\overrightarrow{\gamma}, k)$, find a seed set $G^* = \arg\max_S \sigma(S|\overrightarrow{\gamma})$, where $S \subset \mathcal{V}$, $|S| = k$.*

In the greedy seed selection algorithm, the node with the largest maximal gain can be defined as follows:

Problem 2 *(Seed node selection) In the setting of Problem 1, given a partial solution S', $|S'| < k$, of the greedy seed selection, find the node $s \in \mathcal{V} - S'$ with the largest marginal spread gain w.r.t S', i.e.,* $\arg\max_s [\sigma(S' \cup \{s\}|\overrightarrow{\gamma}) - \sigma(S'|\overrightarrow{\gamma})]$.

Finally, we aim to replace any expensive simulation-based estimation of marginal spread with a prediction thereof, given by a model $\mathcal{M}$ trained on the diffusion graph.

Problem 3 *(Marginal gain prediction) For a known topic-aware network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, build a predictive model $\mathcal{M}$ that– for any query $\mathcal{Q} = (\overrightarrow{\gamma}, k)$ and partial greedy solution S'– predicts the marginal spread gain w.r.t. S' for any node $s \in \mathcal{V} - S'$,*

$$\mathcal{1}(\overrightarrow{\gamma}, S', s) \approx [\sigma(S' \cup \{s\}|\overrightarrow{\gamma}) - \sigma(S'|\overrightarrow{\gamma})] \tag{2}$$

Our focus is on solving in a satisfactory manner Problem (3), s.t. Problem (1) can be answered more efficiently, by substituting the spread σ-expression in Problem (2) with an $\mathcal{M}$-prediction thereof.

4 Our approach

In this section, we present the progression of our models for the Influence Maximization (IM) problem in social networks. We start with the adaptation of the S2V-DQN framework Dai et al. (2017) to IM, which lays the foundation for our subsequent models. We then introduce **IM-GNN**, our advanced Graph Neural Network (GNN) based approach for vanilla IM. Recognizing the importance of topic-awareness in real-world scenarios, we extend our model to **TIM-GNN**. Finally, to enhance efficiency, we develop **TIM-GNNx**, which optimizes TIM-GNN using cross-attention mechanisms. Throughout this progression, we emphasize the key differences and motivations behind each improvement.

4.1 S2V-DQN-IM: a foundational approach

S2V-DQN-IM serves as our initial model, stemming from Dai et al. (2017)'s S2V-DQN framework and being tailored for vanilla IM. This basic version involves train-

ing on sub-graphs extracted from the input diffusion graph, and is followed in PIANO Li et al. (2022). A key component S2V-DQN-IM is a Graph Neural Network (GNN) employing the Structure2Vec (S2V) framework[2]Dai et al. (2016). S2V-DQN-IM incrementally constructs the IM seed set. Each iteration selects a candidate node guided by an action-value function Q, derived from GNN-produced node embeddings. Q quantifies the nodes' potential contribution (marginal spread gain) in the context of the current partial solution.

GNN and node embedding. The GNN architecture directly contributes to the parametrization of the Q-function, denoted as $\hat{Q}$. This involves computing a p-dimensional feature embedding μ_v for each node $v \in \mathcal{V}$, taking into account the partial solution S'. The embedding process is as follows:

$$\mu_{,}^{t+1} \leftarrow relu\left(\theta_1 x_v + \theta_2\left(\sum_{u\in\mathcal{N}(v)} \mu_u^t\right) + \theta_3\left(\sum_{u\in\mathcal{N}(v)} relu(\theta_4 p_{v,u})\right)\right) \tag{3}$$

Here, $\theta_1 \in \mathbb{R}^p$, $\theta_2 \in \mathbb{R}^{p\times p}$, $\theta_3 \in \mathbb{R}^{p\times \frac{p}{2}}$, and $\theta_4 \in \mathbb{R}^{\frac{p}{2}}$ are the *model parameters*, p being the embedding size and $\mu_v^0 = 0$. $\mathcal{N}(v)$ is the set of neighboring nodes of node v in the directed graph, where u is considered a neighbor of v if there exists a directed edge from v to u. *relu* is the rectified linear unit ($relu(z) = max(0, z)$) applied element-wise. x_v is a binary scalar encoding the current partial solution S' such that $x_v = 1$ if $v \in S'$ and $x_v = 0$ otherwise.

Upon completing a number T of message passing iterations within the GNN, as outlined in Eq. (3), we obtain the final node vector embeddings, thereby defining the parametrization as follows:

$$\hat{Q}(S', v; \theta) = \theta_5 relu\left(\left[\theta_6 \sum_{u\in V} \mu_u^{(T)}, \theta_7 \mu_v^{(T)}\right]\right) \tag{4}$$

where $\theta_5, \theta_6, \theta_7 \in \mathbb{R}^{p\times \frac{p}{2}}$, and $[.,.]$ is the concatenation operator. A 2-layer MLP reduces $\hat{Q}$ to one value per node. The model $\mathcal{M}$, comprising the set of parameters $\{\theta_i\}_{i=1}^7$ and the two-layer MLP, is trained in an end-to-end fashion using DRL techniques. More precisely, similar to Dai et al. (2017), and subsequently to Li et al. (2022) and Tian et al. (2020), we use a *Prioritized Double DQN* Schaul et al. (2016) to learn $\hat{Q}$.

Overview of the S2V-DQN-IM framework. We provide a comprehensive overview of S2V-DQN-IM in Fig. 1, which will serve as the basis for illustrating our successive adaptations of this framework, towards ultimately an effective and query-time efficient topic-aware IM solution. The process begins with **Step (1)**, where each

[2]GNNs encompass a range of techniques that leverage topology-based node representation learning via message passing, converting the structural information of nodes in a graph into vector representations, with S2V being one such technique.

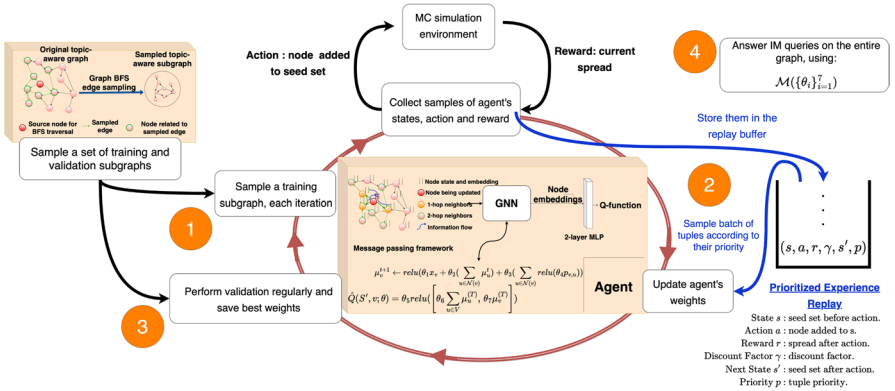

Fig. 1 Overview of S2V-DQN-IM

time a subgraph is sampled[3] from the diffusion graph for training purposes. This strategy serves a dual purpose. First, in alignment with Dai et al. (2017), it enables training on a variety of instances from the combinatorial optimization problem, which are representative of a consistent distribution, thereby enhancing the model's robustness. Second, considering the often very large size of social graphs, this approach aims to address the scalability challenges faced by GNNs when applied to large networks. In **Step (2)**, the agent gathers learning samples from these subgraphs. This is achieved by selecting seed nodes and conducting diffusion simulations to compute the corresponding spread. The accumulated learning samples are stored in a *replay buffer*, which serves as an essential repository during the learning phase of training, contributing significantly to a more efficient use of past experience. The model undergoes continuous validation throughout the learning process (**Step (3)**), where it is tested on larger sub-graph samples. This step is crucial for selecting the most effective model for subsequent evaluation. Finally, in **Step (4)**, the trained model can be used to answer vanilla IM queries. Specifically, it determines an optimal seed set for the entire diffusion graph, for a given query (budget constraint) k. The seed nodes are selected in k successive steps, at each step taking into account the current partial solution.

While S2V-DQN-IM is a valuable starting point for learning-based IM, it has some key limitations for real-world applicability, which our subsequent models aim to rectify.

- *Predictive power* The basic embedding may not capture complex diffusion patterns effectively.
- *Topic-awareness* It does not consider the influence of topics on diffusion, limiting its applicability in real-world settings.
- *Scalability* The foundational framework S2V-DQN struggles with large graphs due to computational complexity.

[3] We use Breadth First Search (BFS) sampling, which starts from a random seed node and systematically explores its neighboring nodes, then their neighbors, and so on.

4.2 IM-GNN: enhancing predictive power

IM-GNN extends S2V-DQN-IM with some advanced GNN features: Graph Attention Mechanisms (GAT) Veličković et al. (2018), positional encodings Geisler et al. (2023), and self-edges. GATs enable the model to discern the significance of node relationships, while positional encodings facilitate the understanding of node positions in the graph, evolving towards a transformer-like architecture. Self-edges can represent a node's inherent properties, which may not be captured otherwise.

Graph attention mechanisms (GAT) Central to the transformer architecture Vaswani et al. (2023), attention mechanisms assign varying importance to neighboring nodes' features.

$$e_{uv}^t = \theta_{att}^T \left[\mu_u^t, \mu_v^t\right] \tag{5}$$

where $\theta_{att} \in \mathbb{R}^{2p}$ is a trainable parameter and $e_{uv}^t \in \mathbb{R}$ indicates the importance of node v's features for node u. Then,

$$\alpha_{uv}^t = \frac{\exp(e_{uv}^t)}{\sum_{v \in \mathcal{N}(u)} \exp(e_{uv}^t)} \tag{6}$$

where α_{uv}^t is the softmax of e_{uv}^t over u's neighbors. The normalized attention weights are used to compute a linear combination of the features of the neighbouring nodes during the message passing aggregation. So Eq. (3) becomes:

$$\leftarrow relu\left(\theta_1 x_v + \theta_2 \left(\sum_{u \in \mathcal{N}(v)} \alpha_{vu}^t \mu_u^t\right) + \theta_3 \left(\sum_{u \in \mathcal{N}(v)} \alpha_{vu}^t relu(\theta_4 p_{v,u})\right)\right) \tag{7}$$

Positional encoding. In IM-GNN, we explore two advanced positional encodings for directed diffusion graphs: the combinatorial Laplacian and the magnetic Laplacian (He et al. 2022). The combinatorial Laplacian, using a symmetrized adjacency matrix, provides insights into the graph's connectivity and structure. The magnetic Laplacian, introducing complex numbers to encode edge direction, gives a more comprehensive representation of directed graphs. This approach aligns conceptually with sinusoidal positional encodings used in transformers Geisler et al. (2023), offering a structure-aware encoding that acknowledges directedness. The degree-normalized version of the magnetic Laplacian is defined as:

$$L_N^{(q)} = I - \left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}\right) \odot \exp\left(i\Theta^{(q)}\right) \tag{8}$$

with the Hadamard product $\odot$, element-wise exp, $i = \sqrt{-1}$, $\Theta_{u,v}^{(q)} = 2\pi q(A_{u,v} - A_{v,u})$, and potential $q \geq 0$.

The magnetic Laplacian is a Hermitian matrix, since $L^{(q)} = (\overline{L}^{(q)})^T$ and, thus, comes with complex eigenvectors. $\Theta_{u,v}^{(q)}$ encodes directedness, s.t., if the graph is

undirected, we recover the combinatorial Laplacian. We get real positional encodings by taking the element-wise module of the eigenvectors. Node embeddings μ_v^0 are then initialized with these encodings (instead of 0).

Self-edges. IM-GNN also integrates self-edges in message passing, which allow the model to retain information about the node's previous state during message passing. This improves the model's predictive capabilities, despite increasing computational demands. Specifically, it entails that $u \in \mathcal{N}(u)$ with $p_{u,u} = 0$ for each node u in the considered graph.

4.3 TIM-GNN: incorporating topic-awareness

In the first topic-aware adaptation of our model, TIM-GNN, we enhance IM-GNN (the version using positional encoding with the magnetic Laplacian) to focus on effectively handling topic-aware scenarios.

Recall from Sect. 3 that, in the topic-aware setting, an IM query is defined as $\mathcal{Q} = (k, \overrightarrow{\gamma})$, where k is the seed set budget and $\overrightarrow{\gamma} \in \mathbb{R}^d$ is the item's topic distribution, with d being the number of topics. Each edge (u, v) in the network has an associated vector of diffusion probabilities $\mathrm{p}_{u,v} \in \mathbb{R}^d$, representing the probability of influence from node u to node v for each topic. Spread is now determined by the TIC diffusion model, and we need to account for this in our approach for predicting marginal spread gain. While the BFS subgraph sampling is common to the two previous approaches, its integration in TIM-GNN requires now *uniform sampling of items* and their projection onto the sampled subgraphs (see Eq. (1)). These projected graphs are then stored and sampled during the training and validation phases.

Referring to Eq. (7), within the trainable parameters of TIM-GNN, θ_3 and θ_4 are specifically related to diffusion probabilities. This new scheme ensures that the model is exposed to a wide variety of topical contexts and that its parameters related to diffusion probabilities are trained such that the model can generalize across the topic space. The other parameters are optimized by the same process as in IM-GNN, benefiting from the fact that only the diffusion probabilities depend on the item's topical distribution, but the topology of the graph remains unchanged.

4.4 TIM-GNN^x: topic-awareness with low query latency

While TIM-GNN models topic-aware diffusion effectively, it requires projecting the item onto the graph at query time, which can be computationally intensive. In the development of **TIM-GNN**^{x}, our objective was to enhance the efficiency of our topic-aware IM method, specifically focusing on *query latency*, without degrading performance. We achieve this by integrating the topic-aware diffusion aspects *directly* into the training process such that, at query time, for any query $\mathcal{Q} = (\overrightarrow{\gamma}, k)$, *the projection of the item* $\overrightarrow{\gamma}$ *onto the diffusion graph is no longer necessary*. In other words, the model will be able to predict marginal spread gain directly based on the query and topic-aware diffusion graph. To obtain this capability, we rely in particular on *cross-attention mechanisms*.

To our knowledge, TIM-GNN^x is the first to address the critical challenges detailed in Sect. 1: predictive *accuracy* over a broad (topic-based) query space, low *latency* over large topic-aware diffusion graphs, *robustness* to graph changes (see the experiments on graph perturbation). The flow of TIM-GNN^x is as follows.

Selection of base items. We begin by creating a set of b *base items*, where $b \geq d$ (recall d is the number of topics). Intuitively, these base items should encapsulate the topical diversity of the existing items. Assuming the set of possible items known (potentially large, but given), one precise and data-driven method to design the base items consists in running a clustering algorithm over the training items (K-Means++ in our experiments), and using the resulting centroids over the training items.

Q-matrix computation: For each base item, we apply TIM-GNN to generate a corresponding Q-function. This process results in a Q-matrix of dimensions $(b, \mathcal{V})$, encapsulating the influence values across different base items and nodes.

Cross-attention mechanism. One key innovation in TIM-GNN^x is the integration of a cross-attention mechanism. This mechanism takes as input the pre-computed Q-matrix (computed once during evaluation) and the specific item $\overrightarrow{\gamma}$ for which a prediction is required, and its role is to dynamically extract the relevant information from the Q-matrix, in order to output a Q-vector specific to $\overrightarrow{\gamma}$.

Training complexity vs. query latency. The attention mechanism is trained end-to-end, trading training complexity for query latency. Indeed, on one hand, by leveraging the summarized information in the Q-matrix, we can make real-time predictions, bypassing the need for expensive graph projection computations at query time. On the other hand, the cross-attention mechanisms and the processing of the base projected graphs bring significant computational overhead and memory requirements. In practice, with limited resources, they may require simplifications in other aspects of the training architecture, potentially leading to a decrease in prediction quality.[4] Figure 2 summarizes the training vs. query time operational differences between TIM-GNN and TIM-GNN^x.

4.5 Model progression summary

We wrap up this section by recalling the progression from IM-GNN to TIM-GNN, and then to TIM-GNN^x, in order to address key challenges in influence maximization:

- **IM-GNN** Enhances predictive power for vanilla IM using advanced GNN features.
- **TIM-GNN** Integrates topic-awareness to handle content-dependent, real-world diffusions.
- **TIM-GNNx** Optimizes query efficiency with cross-attention, enabling fast and accurate topic-aware IM.

[4] In our experimental environment, some of the features introduced in IM-GNN, such as self-edges and positional encoding, had to be silenced, along with a reduction in batch size, embedding dimensions and graph sampling size.

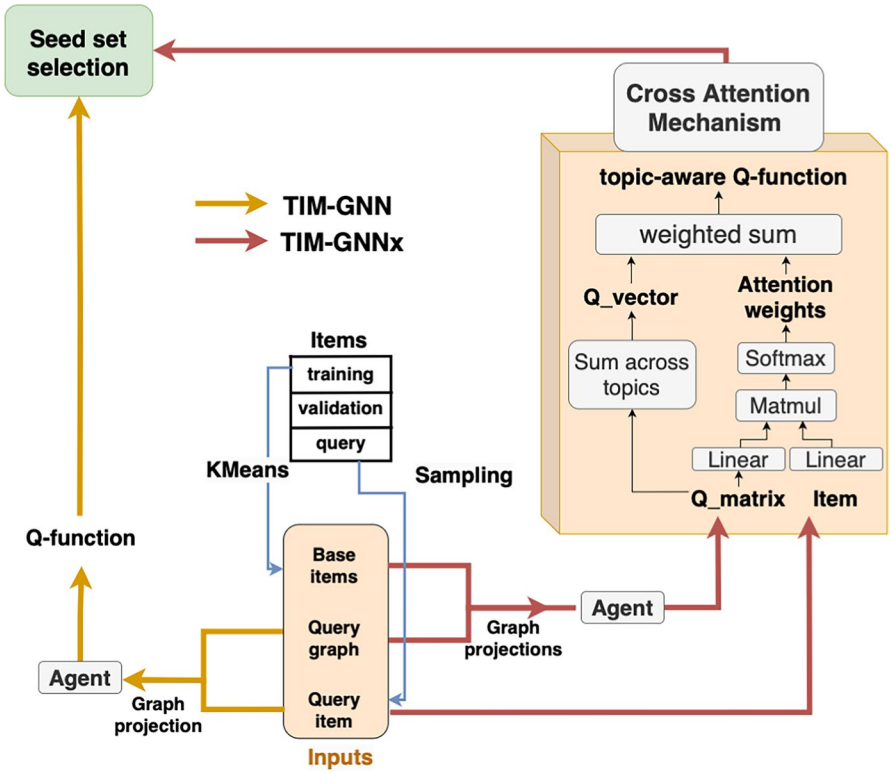

Fig. 2 Comparison TIM-GNN vs. TIM-GNN^x

5 Experiments

In this section, we evaluate the performance of our proposed models on various synthetic and real-world datasets. We analyze the effectiveness (spread), efficiency (query latency), and robustness to changes in the diffusion graph of these methods. We compare our models with baseline algorithms on both topic-agnostic and topic-aware influence maximization tasks.

5.1 Datasets

We conduct experiments on several synthetic and real-world datasets, summarized in Table 1. The datasets are categorized into *topic-agnostic* and *topic-aware* settings. We specify which datasets are used for each variant of our model.

S2V-DQN-IM datasets. Our first goal was to reproduce the direct S2V-DQN framework adaptation to IM; this adaptation was called PIANO in Li et al. (2022). We used the same IM benchmarking datasets **HepPH**, **DBLP**, **LiveJournal**, **Orkut** (all from the SNAP repository); for space reasons, we only report here our results on HepPh, with the other being quite similar. HepPh is a directed graph with 34K nodes and 421K edges. As it is unclear from Li et al. (2022) what diffusion probabilities

Table 1 Summary of datasets

Dataset	Nodes	Edges	Topics	Probabilities	Topics	Kind	Model variant
HepPH	34K	421K	–	Synthetic	No	Directed	S2V-DQN-IM
DBLP	317K	1.05M	–	Synthetic	No	Undirected	S2V-DQN-IM
LiveJournal	4.85M	69 M	–	Synthetic	No	Directed	S2V-DQN-IM
Orkut	3.07M	117.1M	–	Synthetic	No	Undirected	S2V-DQN-IM
Databased Weibo	32K	204K	–	Real-world	No	Directed	IM-GNN
Sina Weibo	61.3K	1.98M	6	Real-world	Yes	Directed	TIM-GNN/TIM-GNN^x
Flixster	22.4K	90.9K	10	Real-world	Yes	Directed	TIM-GNN/TIM-GNN^x
MemeTracker	8.2K	380K	8	Real-world	Yes	Directed	TIM-GNN/TIM-GNN^x

were used for these graphs, we considered either uniform (0.5) or trivalency (0.1, 0.01, 0.001) values.

IM-GNN datasets. IM-GNN was evaluated on realistic diffusion graphs, using a publicly available microblogging dataset from **Sina Weibo** (the major Chinese microblogging platform) Zhang et al. (2013). Starting from the cascades, we employed the edge weighting technique from Goyal et al. (2010) (also used in Panagopoulos et al. (2018)):

$$p(u, v) = (A_{v2u}/A_v) \times e^{-\frac{\bar{D}}{\delta}},$$

where A_{v2u} is the number of times u reposted from v, A_v is the total number of posts of v, and $\bar{D}$ is the average time it takes for u to repost from v. On the resulting graph, we filtered out the edges with probability less than 0.01 and we kept its 5-core. The final graph has 32K nodes and 204K edges.

TIM-GNN & TIM-GNN^x datasets. For the extraction of topic-aware diffusion graphs, besides **Weibo**, we used two other public datasets of diffusion cascades from Barbieri et al. (2017): **Flixster** (a social movie platform where users find and rate movies, connect with friends, discuss, etc) and **MemeTracker clustered** (which tracks phrases and quotes over online-news providers and blogs).

We applied the following pre-process / train / validate / test setup. We **filter** the cascade collection by removing small cascades, short-content, rare users, etc. We fix the number of topics following Barbieri et al. (2017)'s experimental findings. We apply **survival factorization** Barbieri et al. (2017) to infer from the cascades matrices of *authoritativeness* A and *susceptibility* S for the nodes, as well as a matrix of word relevance w.r.t. topics. We get the **topic-aware probabilities** $p_{u,v}$ of our graph by the product of the vectors $A[u]$ and $S[v]$. We create the **items** (topic vectors), using the relevance matrix, by summing the topic distributions for each word in a cascade and normalizing to 1. Once we have the graph and items, we clusted the items by KMeans++, and select the resulting 100 centroids as *representative items*. Among these representative items, we select the 5 items maximizing the Wasserstein distance between the probability distributions of the respective projected graphs, as our **evaluation items**. We also sample 10 items randomly for validation, while the remaining items are used for training.

The final graphs are as follows. Weibo: 61.3K users, 1.98M edges, 6 topics, Flixter: 22.4K users, 90.9K edges, 10 topics, MemeTracker: 8.2K users, 380K edges, 8 topics.

5.2 Baseline methods

We compared with several IM methods, both heuristic-based and learning-based. Besides random selection, **MaxDegree** selects the top k nodes by outdegree, while **MaxOutweight** selects the top k nodes by total weight of their outgoing edges; when probabilities are uniform, this is equivalent to MaxDegree.

ToupleGDD is Chen et al. (2023)'s vanilla IM algorithm; as its result is non-deterministic, this operation is performed 20 times, for statistical relevance, and we take the minimum, mean, and maximum spreads. **GComb** is the vanilla IM algorithm from Manchanda et al. (2020). **DeepIM** is the approach of Ling et al. (2023), solving vanilla IM using an autoencoder to compress seed sets, training the network to optimize the spread prediction from the reconstructed seed set. **S2V-DQN** is Chen et al. (2023)'s implementation of a simple S2V-DQN adaptation for IM, trained on random graphs. **IMM** Tang et al. (2015) is used as the state-of-the-art IM algorithm; its output can be seen as an optimal spread result, to which the heuristics and learning-based methods should get *as close as possible* (i.e., by design, **IMM** will always outperform the other methods in terms of spread). (A detailed discussion on the time and space complexity of our techniques and the baseline methods is provided in Section A).

5.3 Experimental setup

We run our experiments using HPC resources, mainly on (i) Octo-GPU SXM4 80 GB A100 (8-GPU accelerated nodes Nvidia A100 SXM4 80 GB GPUs), with AMD Milan EPYC 7543 processors (32 cores at 2,80 GHz), so 64 cores and 512GB per node, with RedHat8.4 and Slurm21.08.8. Our code is in JAX Bradbury et al. (2018) and Jraph Godwin et al. (2020).[5]

Training. From a given diffusion network, we generate a set of 100 randomly sampled training graphs by BFS. In each training iteration, a new subgraph is sampled, with approximately 20% of the edges from the original network. This allows the model to continuously gather diverse samples of activation states and corresponding rewards, which are then stored in a buffer. Subsequent training phases utilize these buffered samples to update the model's weights.

Validation & testing. We generate another set of 10 graphs with an edge count of 30% of the original graph. We validate the model every 3rd training iteration by evaluating it on these held-out graphs and taking the mean of the spread as a measure of performance. We save the best weights of the model and replace them whenever a better version appears. Finally, the best model selected by validation can be evaluated at query time on the entire graph.

[5] The source code, data pre-processing and statistics, as well as the "glue" code and references to the respective implementations for SurvivalFactorization and baseline methods can be found at https://anonymous.4open.science/r/Submission-ecmlpkdd/.

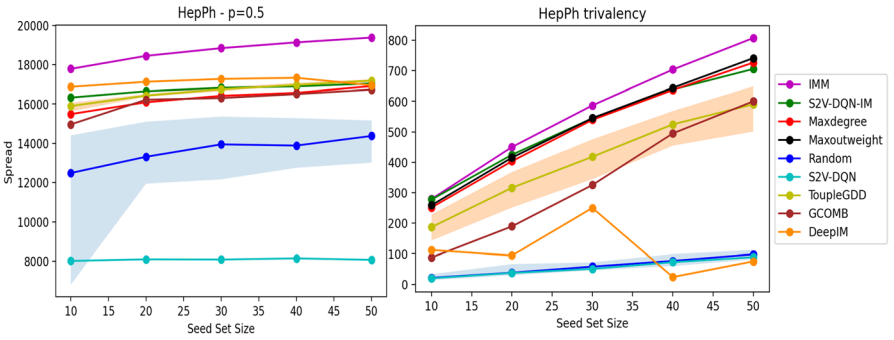

Fig. 3 Spread obtained on HepPh

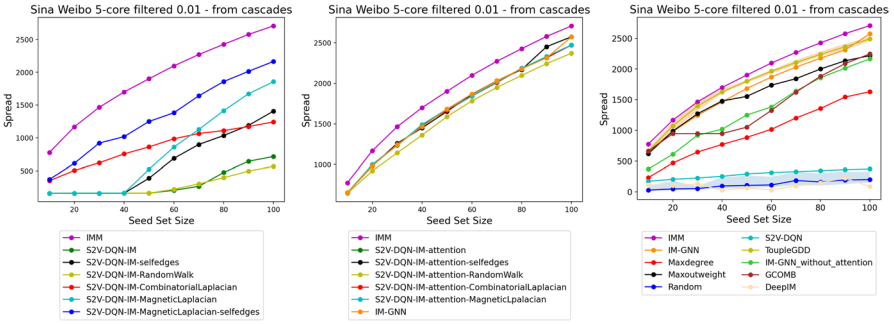

Fig. 4 Spread obtained on Sina Weibo, with variants of S2V-DQN-IM (left), with attention (center), and IM-GNN (right)

5.4 Results and analysis

Evaluation of S2V-DQN-IM. From Fig. 3, we can conclude the following: (1) the performance of S2V-DQN-IM and ToupleGDD are close on uniform (only case where DeepIM also does well) and trivalency graphs, with a slightly better performance for S2V-DQN-IM, and (2) the heuristics MaxDegree/MaxOutweight perform quite well.

Evaluation of IM-GNN. In datasets with uniform or trivalency probability, we have by design a high correlation between the nodes with high degree and those with high total out-weight, hence MaxDegree and MaxOutweight will show similar performance. In realistic diffusion graphs, built from cascades, i.e., with *data-based* diffusion probabilities, this is not necessarily the case.

Recall that IM-GNN stands for S2V-DQN-IM enhanced with attention mechanisms, positional encoding, and self-edges.

In an *ablation study*, we tested all the configurations on the Weibo graph (Fig. 4– therein, we denote the IM-GNN variants by adding to the prefix S2V-DQN-IM the chosen enhancements). The results show that (1) when IM-GNN has no attention, the magnetic Laplacian shows superior performance, while when combined with self-edges, it has a synergistic effect and proves to be very efficient, (2) the combinatorial Laplacian also improves performance compared to the vanilla model, while random-

walks positional encoding hurts the performance, (3) when the attention mechanism is used, it increases the performance but reduces the impact of positional encoding (this may depend on the graph features, so we still consider the magnetic Laplacian and self-edges as useful model additions), (4) as we compare the models with magnetic Laplacian and self-edges, with / without attention, with our baselines, our performance with attention is now above the one of the heuristics (and GComb), comparable to ToupleGDD and, expectedly, below IMM (slightly). ToupleGDD outperforms S2V-DQN-IM on real graphs, perhaps due to its training on random graphs. DeepIM does well (on par with baselines) only on uniform probabilities, but not so well (below baselines) in general.

Evaluation of TIM-GNN and TIM-GNN^x. Recall that, in terms of architecture, TIM-GNN enhances IM-GNN by following a topic-aware training to learn across diverse topic-dependent diffusions.

We evaluate our methods on 15 projected graphs, using the 5 evaluation items on each of the topic-aware graphs. For each projected graph, we compare (1) the baselines, (2) IM-GNN (topic-agnostic model) trained specifically on that same projected graph, (3) TIM-GNN and TIM-GNN^x trained directly in topic-aware manner, with the latter using cross-attention to predict *without item projection onto the topic-aware graph at query time*, and (4) the state-of-the-art INFLEX topic-aware model of Aslay et al. (2014). Recall that INFLEX first. builds an index based on IMM results for 30 random training items; it then aggregates results from this index to answer any topic-aware query.

We present the results in Fig. 5. We can make the following key observations: (1) **(left)** TIM-GNN shows similar performance compared to IM-GNN, competitive performance compared to IMM and INFLEX, and often superior performance compared to ToupleGDD and GComb, (2) MaxOutweight performs surprisingly well on these datasets, (3) **(center)** We observe that TIM-GNN^x generally maintains comparable performance levels to TIM-GNN. It is important to note the performance deviation encountered by TIM-GNN^x on the Sina Weibo dataset. Given the size and complexity of this topic-aware graph, this is due to the additional training complexity and to our current training setting, leading to a loss of critical information necessary for the algorithm to effectively optimize its IM objective[6]; (4) **(right)** when we compare the query latency, we can see TIM-GNN is slower than INFLEX but much faster than ToupleGDD (also an S2V-DQN-based model). Finally, TIM-GNN^x is much faster at query time than TIM-GNN, with a 10x-20x speedup. Moreover, unlike TIM-GNN, which needs a separate prediction for each seed set size k, TIM-GNN^x can predict the Q function just once, allowing for the selection of all seed nodes in a single step. (We also applied the statistical methodology proposed in Demsar (2006) to provide a robust comparison of our methods against the baseline ones, in Section B).

Robustness of TIM-GNN^x. The goal here was to analyze how TIM-GNN^x and INFLEX,[7] the two low-latency topic-aware methods, may be affected by changes in

[6] The model reduction we employed in this dataset was too drastic (embedding size 16 instead 32, batch size 24 instead of 32, graph sampling size 10% instead of 20%).

[7] We only consider INFLEX as a non-learning based method, since all methods of this kind, by design, share its main limitation wrt robustness (in essence, requiring a static graph) due to pre-query topic-aware

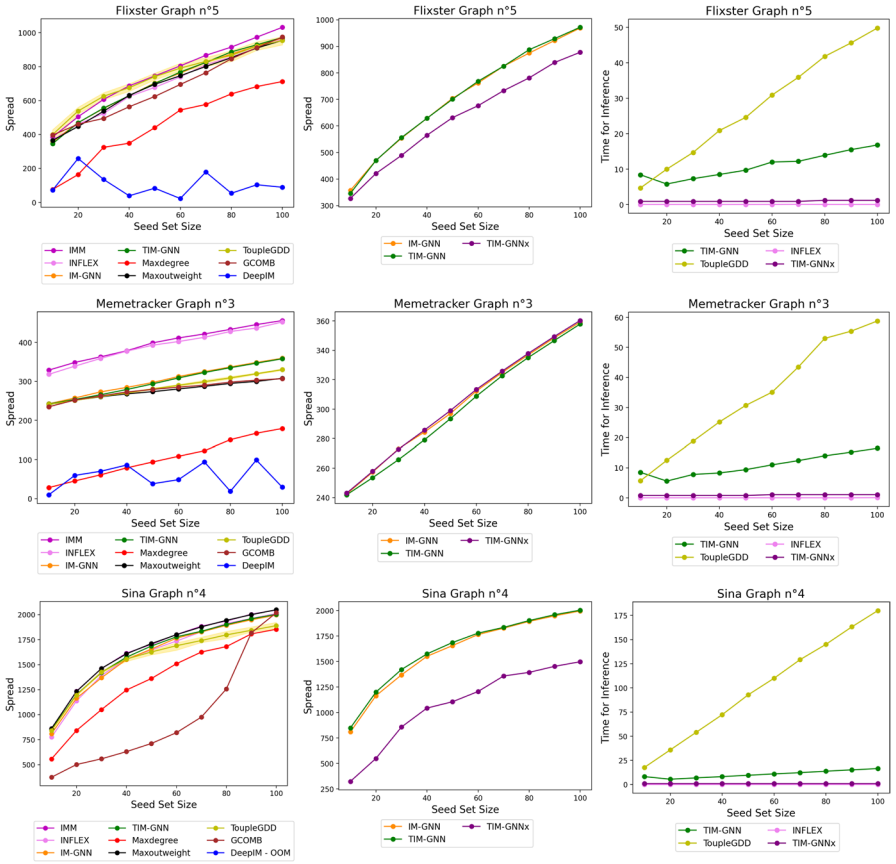

Fig. 5 Results on Flixter (top row), MemeTracker (mid row), and Sina Weibo (bottom row)– Comparative results for a selected evaluation item– TIM-GNN vs. baselines (left), TIM-GNN^x (center), and query latency (right)

diffusion probability values. Our thesis is that INFLEX is not adapted to dynamic diffusion graphs, since it relies on an index, pre-constructed with results from an IM method such as IMM, instead of the actual graph; whenever the graph changes, even slightly, these changes are taken into account by INFLEX only if the index is rebuilt, at a high computation cost.

We performed this experiment for one item per dataset. We first evaluated robustness when facing a limited, *single-node* change, as follows. We initially project the item on the topic-aware input graph, then we select the best node. For INFLEX, the best node is selected from the seed set for $k = 10$, by choosing the one with highest impact on spread. For TIM-GNN^x, since this model provides a ranking, we just choose the node with highest Q-value. We then *perturb* all outgoing probabilities of the chosen node, by setting them to a low value. Recall that, unless the index is

indexing. Note that we do not necessarily aim to do better than INFLEX (or similar methods) on spread (even if we may sometimes), but to achieve a good spread vs. query latency vs. robustness tradeoff.

reconstructed, INFLEX will return the same seed set, regardless of the perturbation. Its performance is therefore evaluated by the spread on the perturbed graph of a seed set obtained from the unperturbed one. We also evaluate robustness for a larger scale, *multi-node* change, which can make the spread of INFLEX arbitrarily low, as follows. Instead of a single node, we select all the nodes from INFLEX's seed set for $k = 100$ and we perturb their outgoing diffusion probabilities to a very small value. The same is done for TIM-GNN^x's 100 best nodes. As a golden standard, we evaluate the spread obtained by IMM on the perturbed graphs in both cases. From Fig. 6, we can observe that, for the single-node perturbation (1*Per*), there is a decrease in performance for both algorithms, but the relative performance remains coherent with our previous results. When we perturb the seed set of INFLEX for $k = 100$ (100*Per*), we can observe that TIM-GNN^x maintains a good performance and stays competitive with IMM, while by design INFLEX has no spread at all besides the seed set.

Time comparison on topic-aware IM. We discuss next execution time aspects. Recall we can distinguish three key time components: training, pre-query processing, query latency. Training time is less critical whenever the model is robust, allowing for infrequent retraining. Pre-query time is also likely manageable, *if not required for each query* (unlike in all the topic-agnostic DRL methods when used in the topic-aware setting, i.e., by *projecting* each time the query topic vector on the topic-aware graph). While non-learning based IM methods like IMM do not have the first 2 stages, they do incur high query latency, which is the main motivation of DRL-based methods: predict marginal spread for faster response time, with a potential loss in spread effectiveness. Recall that Fig. 5, right column, showed query latency; therein IMM was omitted, as it does not even fit the scale.

Finally, we describe *training time* and *memory footprint* for the various DRL-based models. GComb, TGDD, and DeepIM all require < 10h (with some variations depending on the dataset) for training, while TIM-GNN (resp. TIM-GNN^x) < 30

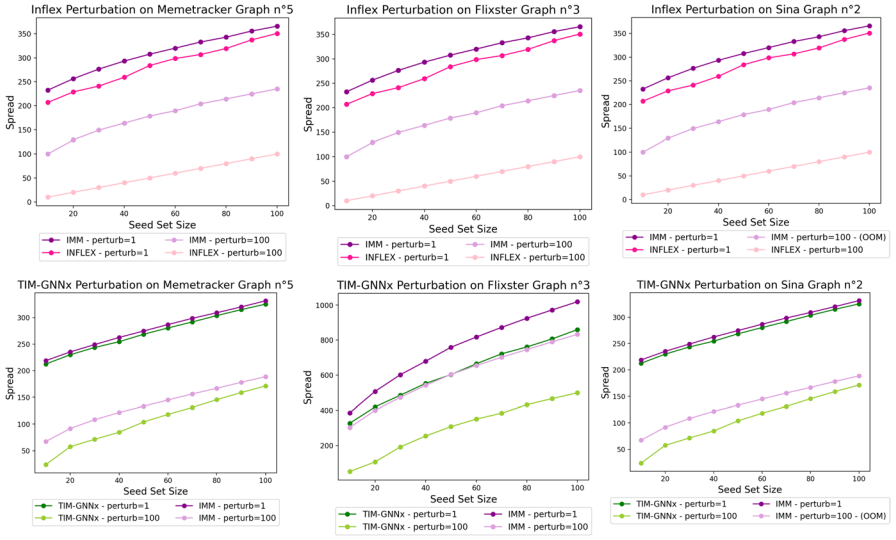

Fig. 6 Perturbation results for INFLEX (top) and TIM-GNN^x (bottom)

h (resp. < 35h). We stress that their larger training time is to be expected, as topic-awareness is integrated in training, but, importantly, this is mitigated by the robustness performance. All models have a < 50GB average memory footprint.

6 Conclusion

We present in this paper DRL-based solutions for IM, building upon the S2V-DQN generic framework, which we progressively refine towards our main contributions, TIM-GNN and TIM-GNN^x, for topic-aware IM. The latter model learns by observing diverse topic-aware diffusions during training. While TIM-GNN performs well in terms of topic-aware spread effectiveness, its latency at query time remains rather high. To address this issue, in TIM-GNN^x, by using cross-attention mechanisms, we integrate the topic-aware dimension directly in the training process. This trades complexity at the training stage for query latency. We show TIM-GNN^x maintains comparable overall spread performance as its predecessor, while achieving a 10x-20x speed-up. To our knowledge, TIM-GNN^x is the first to address the three critical challenges for the practical applicability of learning-based IM methods: predictive accuracy over a broad (topic-based) query space, low latency over large diffusion graphs, and robustness to graph changes, as demonstrated in our empirical evaluation on real-world data and diffusion graphs built from real cascades. By directly predicting topic-aware marginal spread gain, TIM-GNN^x avoids (1) expensive diffusion simulations (as IMM or similar methods), (2) expensive topic / graph projection (as GComb, ToupleGDD, DeepIM, TIM-GNN), and (3) expensive topic-aware indexing, which requires a static graph (as INFLEX). Table 2 summarizes how these methods behave w.r.t. spread, query latency, robustness[8] to graph changes, and topic-awareness.

We identify several potential limitations of our techniques, for application scenarios beyond those from which the experimental data we used originates. First, when the information to be diffused has high dimensionality (e.g., multimedia content), this may impact the latency of TIM-GNN^x, whose number of base items would grow linearly with the dimensionality. One future direction is on adaptations of our techniques for such high-dimensional data. Second, our graph sampling approach relies on the assumption that influence spread is mostly driven by learnable local patterns, that can be captured in moderately sized subgraphs. While this is a reasonable assumption in the diffusion datasets we consider, other applications may exhibit different influence

Table 2 Summary of methods behaviour

	IMM	GComb	ToupleGDD	DeepIM	INFLEX	TIM-GNN	TIM-GNN^x
Spread	✓	✓	✓	✓	✓	✓	✓
Latency	×	×	×	×	✓	×	✓
Scale	×	✓	✓	✓	×	✓	✓
Robust	–	–	–	–	×	✓	✓
Topic-aware	–	–	–	–	✓	✓	✓

[8] Note that robustness and topic-awareness go hand-in-hand, as the common way to deal with topics is to pre-compute and index them away.

dynamics, dominated by global properties or very long-range interactions. In such cases, the performance of models trained on smaller sampled subgraphs could be limited. We leave as a future research direction exploring advanced sampling / training techniques that could bridge local patterns and global structure in GNNs-based approximation algorithms for such combinatorial optimization problems.

A Complexity comparison

Complexity definitions and setup

Let $N = |\mathcal{V}|$ be the number of nodes and $E = |\mathcal{E}|$ the number of edges in the original topic-aware diffusion graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{P})$. Let D be the GNN embedding dimension (default $D = 32$), I the number of GNN message-passing iterations, and k the required seed set size. For topic-aware scenarios, let d be the number of topics and b the number of base items used specifically by TIM-GNN^x for its pre-computation step. For the topic-agnostic diffusion graph obtained by projecting $\mathcal{G}$ onto a specific topic vector $\overrightarrow{\gamma}$ (see Sect. 3), we simply denote its node count as N' and its edge count as E' (where $N' \leq N, E' \leq E$). PARAMS represents the space needed for model parameters.

For the training stage, N_{iter} is the number of training iterations, N_{frames} the steps per iteration, L_p the learning period, B the batch size, mc the number of MC runs per environment step, I_{sim} the average steps per MC simulation (a simple 1D-state propagation GNN = IC model). $N_{total\_frames} = N_{iter} \cdot N_{frames}$, $N_{total\_updates} = N_{total\_frames}/L_p$. We also denote by f the ratio (between 0 and 1) of edges selected in the sampled subgraph during edge sampling.

We structure the complexity discussion along the three stages outlined in the motivation of our paper (Sect. 1), namely training, pre-query, and query / runtime. Depending on the method, training and / or pre-query stages may not apply. When present, the first stage outputs a prediction model. When present, the second stage either combines the diffusion graph and a prediction model (TIM-GNN), or pre-computes and indexes IM results (INFLEX). Finally, at the third stage input queries are handled and ideally should be answered in real-time, with low latency.

Complexity of* TIM-GNN *and* TIM-GNN^x *(Training Stage)

The training phase, while performed offline, represents the bulk of of computational cost required by our DRL approach.

The total training complexity comes from operations repeated over N_{iter} iterations, each containing N_{frames} environment steps and $N_{total\_updates}/N_{iter}$ learning updates:

1. **Subgraph projection (once per iteration):** For TIM-GNN, each iteration starts by sampling an item vector and a subgraph $(N_{subgraph}, E_{subgraph})$ (using edge sampling). As there are N_{iter} iterations, the total cost of those subgraph projections is: $O(N_{iter} \cdot f \cdot E \cdot d)$). There are much fewer iterations than steps so the projection cost is negligible for reasonable values of d.

For TIM-GNN^x, at the beginning of each training iteration, the b base items are projected onto the sampled subgraphs $(N_{subgraph}, E_{subgraph})$ for that iteration.

This generates the base projected subgraphs used for action selection within that iteration. The cost of these projections during training is $O(N_{iter} \cdot b \cdot f \cdot E \cdot d)$. As in TIM-GNN, this cost is incurred less frequently than per-step costs and is typically negligible compared to the overall training cost.

2. **Environment Interaction (per step):** For each of the N_{total_frames} steps:

   - *Action Selection:* To choose an action a_t from state s_t (using ϵ-greedy), the agent must compute Q-values specific to the current query item $\overrightarrow{\gamma_t}$ (and current state s_t).

     For TIM-GNN^x, this involves:

     - Generating the $N_{subgraph} \times b$ base Q-matrix: This requires running the core GNN forward pass b times, once for each base item projected subgraph, derived from state s_t. Cost $O(b \cdot [I \cdot f \cdot E \cdot D + N_{subgraph} \cdot D^2)])$.
     - Applying the cross-attention mechanism using the computed Q-matrix and $\overrightarrow{\gamma_t}$. Cost $O(N_{subgraph} \cdot b \cdot d)$ (+the final argmax step is $O(N_{subgraph})$).

     For **TIM-GNN**, action selection only requires a single GNN pass on the projected graph: $\approx O(I \cdot (f \cdot E \cdot D + N_{subgraph} \cdot D^2))$.

   - *Environment Simulation (per step):* The environment simulates the effect of action a_t using Monte Carlo runs of the IC model. Each step runs *mc* Monte Carlo simulations to estimate the reward. Each simulation takes roughly $O(I_{sim} \cdot f \cdot E)$ time. The total time spent on environment simulation across all training is $O(mc \cdot I_{sim} \cdot f \cdot E)$.

3. **Learning Updates:** Periodically (every L_p steps), the agent samples a batch of B transitions from the replay buffer and performs a gradient update. This requires forward and backward passes through the GNN and the attention module.

For **TIM-GNN**, the cost per update step is dominated by GNN gradient computation (backpropagation has same asymptotic complexity as forward pass): $O(B \cdot [I \cdot (f \cdot E \cdot D + N_{subgraph} \cdot D^2)])$. Total learning time: $O(N_{total\_updates} \cdot B \cdot [I \cdot (f \cdot E \cdot D + N_{subgraph} \cdot D^2)])$.

For **TIM-GNN^x**, for each of the B samples processed, the loss function requires generating the base Q-matrices ($\leq B$, since some transitions may be from the same subgraph). This involves projecting the subgraphs onto the b base items again. This projection cost of $O(b \cdot f \cdot E \cdot d)$ occurs for every sample processed during learning updates, totaling $O(N_{total\_updates} \cdot B \cdot b \cdot f \cdot E \cdot d)$ across training.

The learning update involves gradients through both the GNN (run b times implicitly or explicitly to generate base Q-values) and the attention module. Cost per update: $O(B \cdot [b \cdot I \cdot (f \cdot E \cdot D + N_{subgraph} \cdot D^2) + N_{subgraph} \cdot b \cdot d])$. Total learning time: $O(N_{total\_updates} \cdot B \cdot [b \cdot f \cdot E \cdot d + b \cdot I \cdot (f \cdot E \cdot D + N_{subgraph} \cdot D^2) + N_{subgraph} \cdot b \cdot d])$.

Time Complexity: Sum of projection time, environment interaction and learning update costs over N_{iter} iterations and N_{frames} steps.

For **TIM-GNN^x**:

$$O(N_{iter} \cdot b \cdot f \cdot E \cdot d + N_{total\_frames} \cdot [b \cdot [I \cdot f \cdot E \cdot D + N_{subgraph} \cdot D^2)] + N_{subgraph} \cdot b \cdot d + mc \cdot I_{sim} \cdot f \cdot E] + N_{total\_updates} \cdot B \cdot [b \cdot f \cdot E \cdot d + b \cdot I \cdot (f \cdot E \cdot D + N_{subgraph} \cdot D^2) + N_{subgraph} \cdot b \cdot d])$$

$O(N_{iter} \cdot b \cdot f \cdot E \cdot d + N_{total\_frames} \cdot [b \cdot [I \cdot f \cdot E \cdot D + N_{subgraph} \cdot D^2)] + N_{subgraph} \cdot b \cdot d + mc \cdot I_{sim} \cdot f \cdot E] + N_{total\_updates} \cdot B \cdot [b \cdot f \cdot E \cdot d + b \cdot I \cdot (f \cdot E \cdot D + N_{subgraph} \cdot D^2) + N_{subgraph} \cdot b \cdot d])$.

(dominated by terms involving $b \cdot I \cdot (f \cdot E \cdot D + N_{subgraph} \cdot D^2)$, performed roughly $(N_{total\_frames} + N_{total\_updates} \cdot B)$ times; for small values *d*, $b \leq d$.)

For **TIM-GNN**:

$$O(N_{iter} \cdot f \cdot E \cdot d + N_{total\_frames} \cdot [I \cdot (f \cdot E \cdot D + N_{subgraph} \cdot D^2) + mc \cdot I_{sim} \cdot f \cdot E] + N_{total\_updates} \cdot B \cdot [I \cdot (f \cdot E \cdot D + N_{subgraph} \cdot D^2)])$$

. (dominated by terms involving the GNN computation ($I \cdot (f \cdot E \cdot D + N_{subgraph} \cdot D^2)$), performed roughly $N_{total\_frames} + N_{total\_updates} \cdot B$ times.)

Space Complexity: Dominated by the replay buffer (capacity *M*), model parameters (PARAMS), optimizer states, and activations during backpropagation: $O(M \cdot \text{StateSize} + \text{PARAMS} + B \cdot \text{ActivationSize})$

In the case of TIM-GNN^x we add $O(bN)$ for the Q-matrix.

This highlights that the training phase is computationally demanding, involving both extensive simulation and complex gradient computations, especially for TIM-GNN^x, which includes the factor of *b* in its learning update cost. This offline cost is the primary trade-off for achieving efficiency during the online inference and topic-aware query answering phase.

Complexity of* TIM-GNN *(Pre-query Stage)

No pre-query stage.

Complexity of* TIM-GNN^x *(Pre-query Stage)

At pre-query time, the following is performed once: TIM-GNN^x builds its $b \times N$ Q-matrix by running a model akin to TIM-GNN independently for each of the chosen $b \leq d$ base items. This requires projecting the full graph (*N*, *E*) onto each base item vector. The total projection cost within this stage is $O(b \cdot E \cdot d)$. Subsequently, *b* GNN runs are performed. The total GNN computation cost is $O(b \cdot [I \cdot (E \cdot D + N \cdot D^2)])$. The Q-values for all nodes under each base item are computed and stored.

Time complexity: $O(b \cdot E \cdot d + b \cdot [I \cdot (E \cdot D + N \cdot D^2)])$. This is dominated by the *b* GNN runs, as the projection cost per run ($O(E \cdot d)$) is typically much smaller than the GNN cost ($O(I \cdot (E \cdot D + N \cdot D^2))$).

Table 3 Time and space complexity of all algorithms, for the training, pre-query, and query stages

Algorithm	Training time (assuming small d and $b \leq d$)	
TIM-GNN	$O((N_{total\_frames} + N_{total\_updates} \cdot B) \cdot I \cdot (f \cdot E \cdot D + N_{subgraph} \cdot D^2))$	
TIM-GNN^x	$O((N_{total\_frames} + N_{total\_updates} \cdot B) \cdot b \cdot I \cdot (f \cdot E \cdot D + N_{subgraph} \cdot D^2))$	
GCOMB	Not explicitly stated	
ToupleGDD	$O(N \cdot E + N_{episodes} \cdot N_{frames} \cdot I \cdot (N + E))$	
Algorithm	Training space	
TIM-GNN	$O(M \cdot \text{StateSize} + \text{PARAMS} + B \cdot \text{ActivationSize})$	
TIM-GNN^x	$O(M \cdot \text{StateSize} + \text{PARAMS} + B \cdot \text{ActivationSize} + bN)$	
GCOMB	Not explicitly stated	
ToupleGDD	Not explicitly stated	
Algorithm	Pre-query time	Pre-Query space
TIM-GNN^x	$O(b \cdot [I \cdot (E \cdot D + N \cdot D^2)])$	$O((N + E) \cdot D + \text{PARAMS} + b \cdot N)$
INFLEX	$O(h \cdot T_{IMM})$	$O(T + (t's)^2)$
Algorithm	Query time	Query space
TIM-GNN	$O(E \cdot d + I \cdot (E \cdot D + N \cdot D^2) + N \log k)$	$O((N + E) \cdot D + \text{PARAMS})$
TIM-GNN^x	$O(N \cdot b \cdot d + N \log k)$	$O((N + E) \cdot D + \text{PARAMS} + b \cdot N)$
Greedy	$O(k \cdot N \cdot R \cdot E)$	$O(N + E)$
IMM	$O((k + l)(N_p + E_p) \log N_p / \epsilon^2)$	$O((k + l)(N + E) \log N / \epsilon^2)$
INFLEX	$O(T_{search} + T_{aggregation})$	$O(T + (t's)^2)$
GCOMB	$O(N + \lvert V^{g,I} \rvert (am_g + m_g^2) + \lvert V^g \rvert k(a + m_Q))$	$O(N + E + Im_g^2 + m_Q)$
ToupleGDD	Not explicitly stated, likely dominated by GNN pass $O(I \cdot (N + E))$	Not explicitly stated

Space complexity: The main persistent space cost added by this stage is $O(N \cdot b)$ for the Q-matrix. The peak transient memory is dominated by a single TIM-GNN run ($\approx O((N + E) \cdot D + \text{PARAMS})$).

Complexity of* TIM-GNN *(Runtime / Query Stage)

This is the phase where the trained model is used to select the seed set for a given query (d-dimensional topic-vector and seed size k) $\mathcal{Q} = (\overrightarrow{\gamma}, k)$. The following steps are taken at query-time:

1. From the original topic-aware diffusion graph $\mathcal{G}$, a topic-agnostic graph (N, E) is obtained by projecting onto the query vector $\overrightarrow{\gamma}$. This projection cost is $O(E \cdot d)$ and is performed once per query, preceding the main computation.
2. We run the trained GNN on the projected graph (N, E) to compute Q-value estimates. The cost is $O(I \cdot (E \cdot D + N \cdot D^2))$.
3. We select the top k nodes based on these Q-values. The cost is $O(N \log k)$.

Time complexity (per query): $O(E \cdot d + I \cdot (E \cdot D + N \cdot D^2) + N \log k)$. While the projection cost $O(E \cdot d)$ is present, it is often negligible compared to the GNN term.

Space Complexity: Dominated by keeping the embeddings during the forward pass and the model parameters: $O((N + E) \cdot D + \text{PARAMS})$.

Table 4 Parameters in complexity analysis and typical orders of magnitude

Symbol	Description	Typical Order/Value/Relationship
N	Number of nodes	Large, experiments: 10^4
E	Number of edges	Large, experiments: $10^4 - 10^6$
D	GNN embedding dimension	Small constant, experiments: 16, 32, 64
I	GNN iterations	Small constant, experiments: 5
PARAMS	GNN model parameters space	Depends on layers and D^2
k	Seed set size	Small to moderate, experiments: 10 - 100
d	Topic dimension	Small, experiments: 6-10
b	Number of base items	Paper: $b = d$; Potentially $b \ll d$ for high-dimension topics
h	Number of index items	Paper: 30
ϵ	IMM approximation error	Small constant, experiments: 0.1
l	INFLEX parameter	Small constant, experiments: 5
s	INFLEX precomputed list size	Similar to k (experiments: 100)
t'	INFLEX neighbors retrieved	Small constant, experiments: $\leq$ 10
N_{iter}	Number of training iterations	Moderate, experiments: 5 - 15
N_{frames}	Env. steps per iteration	Moderate, experiments: 160
N_{total_frames}	Total env. steps	Large ($N_{iter} \times N_{frames}$)
L_p	Learning period	Small constant, experiments: 20
$N_{total\_updates}$	Total learning updates	Moderate (N_{total_frames}/L_p)
B	Batch size	Moderate, experiments: 16, 32, 64
mc	MC runs per env. step	Moderate constant, experiments: 100 for train, 1000 for eval
I_{sim}	Average steps per MC simulation	Graph dependent (related to diffusion depth/diameter)
M	Replay buffer capacity	Large, experiments: 10^6
StateSize	Size of stored state in buffer	Stored: seed set lists (at t-1 and t), reward, action, discount
ActivationSize	Size of activations per sample	Depends on N, E, D, I (can be large for backprop)

Complexity of* TIM-GNN^x *(Runtime / Query Stage)

For any input query $(\overrightarrow{\gamma}, k)$, TIM-GNN^x completely **avoids graph projection**. It applies a fast cross-attention mechanism (C_{attn}) that uses the d-dimensional query vector $\overrightarrow{\gamma}$ and the pre-computed $b \times N$ Q-matrix to generate item-specific Q-values for all N nodes. This is followed by top-k selection ($O(N \log k)$). The cross-attention

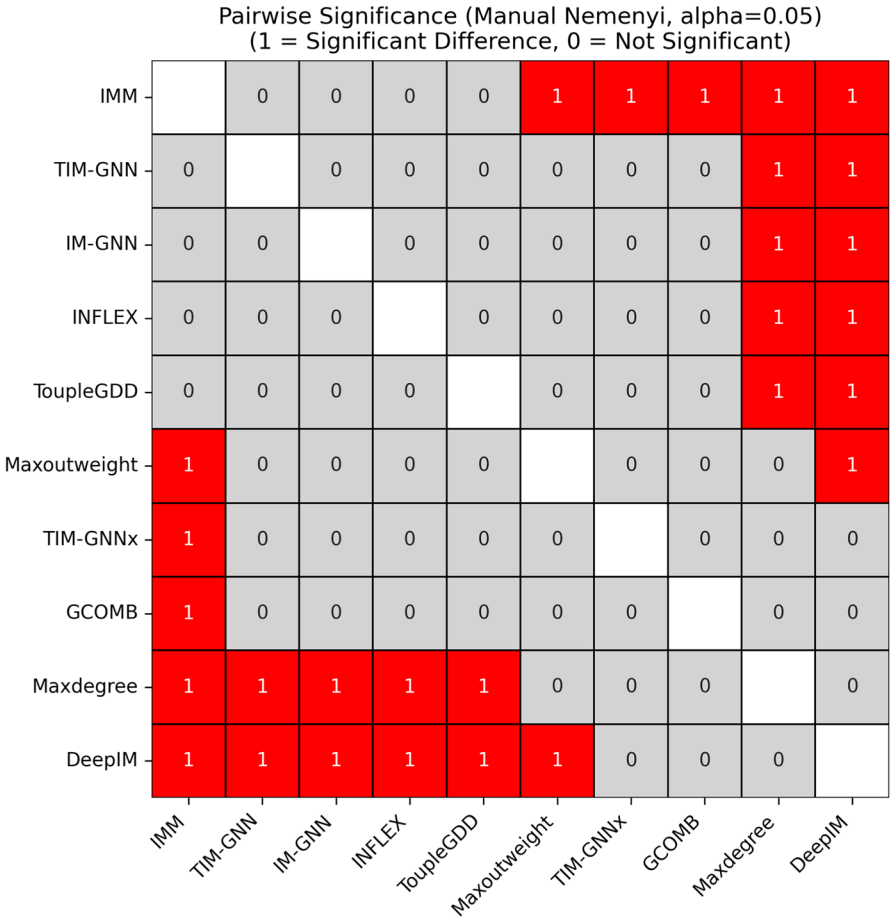

Fig. 7 Heatmap of pairwise significance based on the Nemenyi test ($\alpha = 0.05$), applied to average ranks of spread at k=50. Red (1) indicates a statistically significant difference for a pair, while Grey (0) indicates no significant difference

cost $C_{attn} = O(N \cdot b \cdot d)$ arises primarily from operations (applying linear layers) involving the Q-matrix and the query vector.

Time complexity (per query): $O(N \cdot b \cdot d + N \log k)$. This is significantly faster than a full GNN pass.

Space Complexity: Dominated by GNN parameters and the stored Q-matrix: $\approx O((N + E) \cdot D + \text{PARAMS} + b \cdot N)$.

Importantly, neither TIM-GNN nor TIM-GNN^x relies on iterative Monte Carlo (MC) simulations or the generation/processing of large numbers of Reverse Reachable (RR) sets at query time, during the online seed selection step(s).

Complexity comparison with baseline methods.

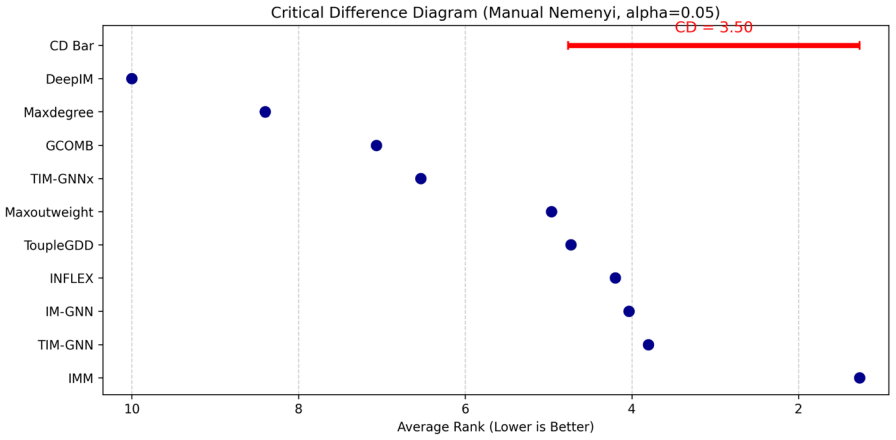

Fig. 8 Critical Difference (CD) diagram based on the Nemenyi test results

Let ϵ be the IMM approximation error term, l is an IMM parameter related to failure probability (usually O(1)), h the number of indexed items for INFLEX, and s the INFLEX list size.

Recall that traditional heuristic or learning-based IM methods, when adapted for topic-aware IM, require running their topic-agnostic IM algorithm on the projected graph, i.e., the topic-agnostic diffusion graph obtained by projecting $\mathcal{G}$ onto a specific topic vector $\overrightarrow{\gamma}$. We will not repeat the cost of this projection step in what follows, as it is incurred by all the methods (except INFLEX).

Only the learning-based methods have a training stage. INFLEX has a pre-query stage, where it computes and indexes the results of certain topic-aware queries.

- **Complexity of Greedy (Runtime / Query Stage)** Batch algorithm solving a single IM instance, providing a $(1 - 1/e - \varepsilon)$ approximation guarantee (where ε often relates to estimation error). The basic Greedy algorithm iteratively finds the node with the highest marginal influence gain.

  Time Complexity: $O(k \cdot N \cdot R \cdot E)$ using R MC simulations.
  Space Complexity: Mainly storing the graph, $O(N + E)$, plus structures for the optimization (e.g., priority queue $O(N)$).

- **Complexity of IMM** Tang et al. (2015) **(Runtime / Query Stage)** A batch algorithm designed to solve a single instance of the topic-agnostic IM problem for a given graph, diffusion model, seed set size k, and approximation parameter ε. Provides strong theoretical guarantees, returning a $(1 - 1/e - \varepsilon)$ approximate solution with high probability.

  Time complexity: $O((k + l)(N_p + E_p) \log N_p/\epsilon^2)$.
  Space complexity: Dominated by storing RR sets, $O((k + l)(N + E) \log N/\epsilon^2)$.

- **Complexity of INFLEX** Aslay et al. (2014) **(Pre-query Stage)** This topic-aware

index-based method requires a projection for each of the *h* indexing queries, using topic-agnostic IMM.

Time complexity: $O(h \cdot T_{IMM})$, where T_{IMM} denotes the IMM runtime complexity.
Space complexity: $O(E \cdot T + h(T + s))$. Represents the permanent storage for the precomputed index, including base topic probabilities ($E \cdot T$), the *h* index points ($h \cdot T$), their precomputed *s*-sized seed lists ($h \cdot s$), and the bb-tree structure.

- **Complexity of INFLEX (Runtime / Query Stage)** Achieved by combining an efficient approximate nearest neighbor search (T_search) within the precomputed index with a lightweight rank aggregation (T_aggregation) of the retrieved neighbor seed lists.

  Time complexity: $O(T_{search} + T_{aggregation})$.
  Space complexity: $O(T + (t's)^2)$. Represents the small, temporary working memory needed per query, holding the query vector (*T*), the t' retrieved seed lists ($t's$), and data structures for rank aggregation (up to $O((t's)^2)$).

- **Complexity of GCOMB** Manchanda et al. (2020) **(Runtime / Query Stage)** A topic-agnostic learning-based approach using GCN and Q-learning. Let m_g be GCN embedding dimension, m_Q be Q-learning embedding dimension, *I* be number of GCN layers, $|V^g|$ be the set of "good quality" nodes identified via the noise predictor, $|V^{g,K}|$ be the K-hop neighborhood of good nodes. *a* is average degree. *k* seed set size. No explicit complexity in O-notation provided in the paper for training. No pre-query stage.

  Time Complexity: $\approx O(N + |V^{g,I}|(am_g + m_g^2) + |V^g|k(a + m_Q))$. (Ignoring the minor $O(N \log l)$ term for the node noise predictor).
  Space Complexity: $O(N + E + Im_g^2 + m_Q)$.

- **Complexity of ToupleGDD** Chen et al. (2023) **(Training Stage)** A topic-agnostic learning-based approach using coupled GNNs and Double DQN. Let $N_{episodes}$ be the number of training episodes, *I* be the number of GNN layers.

  Time complexity: $O(N \cdot E + N_{episodes} \cdot N_{frames} \cdot I \cdot (N + E))$. (Includes $O(N \cdot E)$ for Personalized DeepWalk pre-step).
  Space Complexity: Not explicitly stated in O-notation in the paper. Involves mostly the graph encoding, parameters, embeddings, and a replay buffer.

- **Complexity of ToupleGDD (Runtime / Query Stage)**

  Time Complexity: Not explicitly stated in O-notation in the paper. Empirical results show fast inference (order of seconds). One-time inference likely dominated by

GNN pass $O(I \cdot (N + E))$. Iterative inference complexity depends on recomputation strategy (not specified).
Space Complexity: Not explicitly stated in O-notation in the paper. Involves mostly the graph encoding, parameters, and embeddings.

Scalability & trade-offs discussion (motivation of the* TIM-GNN^x *approach)
The detailed complexity analysis across the training, pre-query, and query stages allows for a quantitative assessment of the scalability and practical interest of our proposed methods, particularly TIM-GNN^x, compared to TIM-GNN and the baseline methods. The main focus is on the trade-off between offline computation cost and online (query time) performance (aiming for real-time, low latency and high spread).

- **Offline cost vs. online performance.** Traditional methods like Greedy / CELF and IMM perform most computations at query time, leading to high latency. INFLEX transfers most of this cost to the offline stage, achieving low latency, at the cost of very high pre-computation (running IMM *h* times– order of days in our experiments) and **limited robustness**. Our GNN-based methods strike a better balance between these objectives:
  - TIM-GNN requires intensive offline training but is relatively fast at query time and yields high spread, although it necessitates per-query projection of the topic-aware diffusion graph.
  - TIM-GNN^x has the most intensive offline phase, involving both training and a one-time Q-matrix pre-computation. In return, it offers extremely fast, projection-free online queries ($O(Nbd + N \log k)$) with spread quality that is almost as good as the one of TIM-GNN.

- **TIM-GNN vs. TIM-GNN^x- the graph projection bottleneck.** TIM-GNN, while having a simpler training regime than TIM-GNN^x, requires a graph projection step plus a full GNN pass for *every* topic-specific query. TIM-GNN^x is designed to avoid such per-query costs. Its fast attention query ($O(Nbd)$) performed directly on the pre-computed Q-matrix makes it significantly faster than TIM-GNN for topic-aware queries, justifying its increased offline complexity whenever low latency across many topics is required. In conclusion, the two techniques are complementary, and one or the other may be preferred, depending on the required tradeoff.
- **TIM-GNN^x vs. INFLEX - pre-computation costs.** The pre-query cost of INFLEX involves potentially hundreds of runs of slow algorithms like IMM / CELF++. TIM-GNN^x's pre-computation, while significant, relies on the faster GNN inference runs and is more practical. Furthermore, the learned nature of TIM-GNN^x's components (GNN, attention) offer much better robustness to graph changes than INFLEX's static index, which would necessitate a full, costly rebuild.
- **Space complexity comparison:** TIM-GNN has modest space needs ($O((N + E)D + \text{PARAMS})$). TIM-GNNx adds the Q-matrix ($O(Nb)$). In conclusion TIM-GNN^x is designed specifically for applications that require:

1. Topic-specific spread queries (in our view, most social media scenarios).
2. Very low query latency, significantly faster than TIM-GNN, IMM, or Greedy can provide (in our view, a key requirement for practical purposes).
3. Manageable pre-query / storage costs (which the SOTA topic-aware IM method INFLEX does not provide).
4. Robustness to graph changes (necessary to avoid costly re-training, which the SOTA topic-aware IM method INFLEX does not provide).We summarize the various costs of all methods in Table 3, while Table 4 recalls the main cost ingredients and their order in magnitude in our experiments.

B Statistical analysis

We follow here the statistical methodology proposed by Demsar (2006) to provide a robust comparison of our proposed methods (TIM-GNN, TIM-GNN^x) against the baseline methods.

Specifically, we have performed the following steps.

1. **Performance metric selection:** We used the influence spread achieved by each algorithm at a mid-range seed set size, specifically **k=50**, as the primary performance metric for this statistical comparison. We chose k=50 as it represents performance with a moderate budget (our seed set ranged from 10 to 100 in the experiments).
2. **Ranking procedure:** We evaluated performance across 15 distinct test cases (5 evaluation items/projections for each of the 3 topic-aware datasets– Sina Weibo, Flixster, MemeTracker). For each case, we ranked the 10 compared algorithms based on their achieved spread at k=50 (Rank 1 = Best). Ties were handled using average ranks. We then calculated the average rank for each algorithm across all 15 cases.
3. **Overall comparison (Friedman test):** We applied the non-parametric **Friedman test** to these ranks. The test yielded a statistic of **94.54** with a p-value of $\mathbf{p \approx 0.0000}$. Since the p-value is less than our significance level of $\alpha = 0.05$, we reject the null hypothesis and conclude that statistically significant differences exist among the algorithms' overall performance based on average rank.
4. **Pairwise comparison (Nemenyi procedure):** Following the significant Friedman test, we conducted the **Nemenyi post-hoc comparison** at $\alpha = 0.05$. For 10 algorithms and $N = 15$ datasets, the critical value $q_{0.05}$ is 3.164. This yields a **Critical Difference (CD) of 3.4979**. Pairs of algorithms whose average ranks differ by more than this CD are considered statistically significantly different (see Fig. 7).
5. **Visualization (CD Diagram):** We generated a **Critical Difference (CD) diagram** (Fig. 8) based on the Nemenyi test results. This diagram (Fig. 8) presents the average ranks (lower is better) and connects groups of algorithms that are statistically indistinguishable (rank difference $\leq$ CD).

Interpretation of statistical results

The analysis reveals the following insights based on average ranks at k=50:

- **Top performers:** As expected, IMM achieves the best average rank (1.26). Recall that in our setting IMM can be seen as the golden stardard, to which the other methods must get as close as possible in terms of spread performance; IMM's high query-time costs precludes its usage in practical topic-aware scenarios. However, based on the Nemenyi test (CD = 3.4979), its performance is statistically indistinguishable from TIM-GNN (3.80), IM-GNN (4.03), INFLEX (4.20) and ToupleGDD (4.73). These algorithms form the top-performing group.
- **Our models' performance:** TIM-GNN ranks well within the top group, statistically similar to IMM and INFLEX. TIM-GNN^x (average rank 6.53) performs significantly worse than IMM but is statistically indistinguishable from the group including TIM-GNN, IM-GNN, INFLEX, ToupleGDD, Maxoutweight, GCOMB, Maxdegree and even DeepIM. (This ranking reflects performance solely based on spread at k=50, and does not account for TIM-GNN^x's other significant advantages, namely latency and robustness).
- **Other baselines:** MaxOutWeight performs surprisingly well, statistically similar to the top group (except IMM). GCOMB, Maxdegree, and DeepIM occupy the lower ranks, with DeepIM being significantly outperformed by most other methods.

C Motivation for DRL

We discuss in this section in more detail what makes DRL a compelling learning approach in our problem setting.

The primary motivation for training-based testems from the limitations of traditional IM methods. The standard greedy algorithm, while providing approximation guarantees, is computationally expensive due to the need for repeated Monte Carlo (MC) simulations or RR-sets to estimate marginal influence gains (a #P-hard problem). This makes it unusable in practice, for reasonably large graphs and for low-latency requirements.

Deep Reinforcement Learning (DRL), particularly using Q-learning with Graph Neural Networks (GNNs) as function approximators (building on the S2V-DQN framework Dai et al. (2017)), offers a compelling alternative. It frames the sequential seed selection process as a Markov Decision Process (MDP) and aims to learn an action-value function $Q(S', v)$. This function approximates the expected total future spread resulting from adding node v to the current seed set S'.

The key advantages of this DRL approach are as follows:

1. **Efficiency at inference:** By learning the Q-function offline (using MC simulations within the training environment), the computationally expensive step is shifted away from the online inference phase. Seed selection during inference reduces to efficiently querying the learned Q-function (via a GNN forward pass) and selecting the best node at each step, replacing costly online MC simulations.

2. **Leveraging the graph structure:** GNNs naturally process graph structures and node / edge features, enabling the Q-function to learn complex influence patterns and dependencies more effectively than simple heuristics.We stress that our contribution lies not just in applying DRL, but in significantly enhancing the foundational S2V-DQN framework for the complexity and requirements of real-world IM tasks:

- We employ advanced GNN features (attention, positional encodings) for improved predictive accuracy over basic implementations (IM-GNN basis).
- We develop novel mechanisms to handle topic-aware influence effectively (TIM-GNN's training strategy and TIM-GNN^x's cross-attention mechanism), addressing a key limitation of prior vanilla IM models. TIM-GNN^x specifically optimizes for low-latency topic-aware queries by eliminating the need for per-query graph projection at query time.In conclusion, RL provides in this problem setting a principled way to learn an efficient approximation model for the computationally hard greedy selection strategy.

Acknowledgements This research is part of the programme DesCartes and is supported by the National-Research Foundation, Prime Minister's Office, Singapore under its Campus for ResearchExcellence and Technological Enterprise (CREATE) programme.

Funding Open access funding provided by Université Paris-Saclay.

References

Arora A, Galhotra S, Ranu S (2017) Debunking the myths of influence maximization: An in-depth benchmarking study. In: Salihoglu S, Zhou W, Chirkova R, Yang J, Suciu D (eds.) Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017, pp. 651–666. ACM. https://doi.org/10.1145/3035918.3035924

Aslay Ç, Barbieri N, Bonchi F, Baeza-Yates R (2014) Online topic-aware influence maximization queries. In: Amer-Yahia S, Christophides V, Kementsietsidis A, Garofalakis MN, Idreos S, Leroy V (eds.) Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, March 24-28, 2014, pp. 295–306. OpenProceedings.org. https://doi.org/10.5441/002/edbt.2014.28

Barbieri N, Bonchi F, Manco G (2013) Topic-aware social influence propagation models. Knowl Inf Syst 37(3):555–584. https://doi.org/10.1007/s10115-013-0646-6

Barbieri N, Manco G, Ritacco E (2017) Survival factorization on diffusion networks. In: Ceci M, Hollmén J, Todorovski L, Vens C, Džeroski S (eds) Machine learning and knowledge discovery in databases. Springer, Cham, pp 684–700

Borgs C, Brautbar M, Chayes JT, Lucier B (2012) Influence maximization in social networks: Towards an optimal algorithmic solution. CoRR arXiv:abs/1212.0884

Bradbury J, Frostig R, Hawkins P, Johnson MJ, Leary C, Maclaurin D, Necula G, Paszke A, VanderPlas J, Wanderman-Milne S, Zhang Q (2018) JAX: Composable Transformations of Python+NumPy programs. http://github.com/google/jax

Chen S, Fan J, Li G, Feng J, Tan K, Tang J (2015b) Online topic-aware influence maximization. Proc VLDB Endow 8(6):666–677

Chen T, Yan S, Guo J, Wu W (2023) ToupleGDD: a fine-designed solution of influence maximization by deep reinforcement learning. IEEE Trans Comput Soc Syst. https://doi.org/10.1109/tcss.2023.3272331

Chen W, Lin T, Yang C (2015a) Real-time topic-aware influence maximization using preprocessing. In: Thai MT, Nguyen NP, Shen H (eds.) Computational Social Networks - 4th International Conference, CSoNet 2015, Beijing, China, August 4-6, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9197, pp. 1–13. Springer. https://doi.org/10.1007/978-3-319-21786-4_1

Chen H, Qiu W, Ou H, An B, Tambe M (2021) Contingency-aware influence maximization: a reinforcement learning approach. In: UAI

Dai H, Dai B, Song L (2016) Discriminative embeddings of latent variable models for structured data. CoRR arXiv:abs/1603.05629

Dai H, Khalil EB, Zhang Y, Dilkina B, Song L (2017) Learning combinatorial optimization algorithms over graphs. CoRR arXiv:abs/1704.01665

Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Du N, Song L, Woo H, Zha H (2013) Uncover topic-sensitive information diffusion networks. In: Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013, pp. 229–237

Geisler S, Li Y, Mankowitz D, Cemgil AT, Günnemann S, Paduraru C (2023) Transformers Meet Directed Graphs

Godwin J, Keck T, Battaglia P, Bapst V, Kipf T, Li Y, Stachenfeld K, Veličković P, Sanchez-Gonzalez A (2020) Jraph: a library for graph neural networks in jax. http://github.com/deepmind/jraph

Goyal A, Bonchi F, Lakshmanan LVS (2010) Learning influence probabilities in social networks. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining. WSDM '10, pp. 241–250. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1718487.1718518

He Y, Permultter M, Reinert G, Cucuringu M (2022) MSGNN: a spectral graph neural network based on a novel magnetic signed Laplacian

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 '03, pp. 137–146. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/956750.956769

Leskovec J, Krause A, Guestrin C, Faloutsos C, VanBriesen J, Glance N (2007) Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '07, pp. 420–429. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1281192.1281239

Li Y, Fan J, Wang Y, Tan K (2018) Influence maximization on social graphs: a survey. IEEE Trans Knowl Data Eng 30(10):1852–1872. https://doi.org/10.1109/TKDE.2018.2807843

Li Y, Gao H, Gao Y, Guo J, Wu W (2023) A survey on influence maximization: from an ml-based combinatorial optimization. ACM Trans Knowl Discov Data. https://doi.org/10.1145/3604559

Li H, Xu M, Bhowmick SS, Rayhan JS, Sun C, Cui J (2022) Piano: influence maximization meets deep reinforcement learning. IEEE Trans Comput Soc Syst. https://doi.org/10.1109/TCSS.2022.3164667

Ling C, Jiang J, Wang J, Thai MT, Xue R, Song J, Qiu M, Zhao L (2023) Deep graph representation learning and optimization for influence maximization. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA. Proceedings of Machine Learning Research, vol. 202, pp. 21350–21361. PMLR. https://proceedings.mlr.press/v202/ling23b.html

Manchanda S, Mittal A, Dhawan A, Medya S, Ranu S, Singh A (2020) Gcomb: learning budget-constrained combinatorial algorithms over billion-sized graphs. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 20000–20011. Curran Associates, Inc.

Ohsaka N (2020) The solution distribution of influence maximization: a high-level experimental study on three algorithmic approaches. In: Maier D, Pottinger R, Doan A, Tan W, Alawini A, Ngo HQ (eds.) Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, Online Conference [Portland, OR, USA], June 14-19, 2020, pp. 2151–2166. ACM. https://doi.org/10.1145/3318464.3380564

Panagopoulos G, Malliaros FD, Vazirgiannis M (2022) Multi-task learning for influence estimation and maximization. IEEE Trans Knowl Data Eng 34(9):4398–4409

Panagopoulos G, Malliaros FD, Vazirgiannis M (2018) Diffugreedy: An influence maximization algorithm based on diffusion cascades. In: Aiello LM, Cherifi C, Cherifi H, Lambiotte R, Lió P, Rocha LM (eds.) Complex Networks and Their Applications VII - Volume 1 Proceedings The 7th International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2018, Cambridge, UK, December 11-13, 2018. Studies in Computational Intelligence, vol. 812, pp. 392–404. Springer. https://doi.org/10.1007/978-3-030-05411-3_32

Schaul T, Quan J, Antonoglou I, Silver D (2016) Prioritized Experience Replay

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 '15, pp. 1539–1554. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2723372.2723734

Tian S, Mo S, Wang L, Peng Z (2020) Deep reinforcement learning-based approach to tackle topic-aware influence maximization. Data Sci Eng 5(1):1–11

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2023) Attention is all you need

Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph Attention Networks

Xia W, Li Y, Wu J, Li S (2021) Deepis: Susceptibility estimation on social networks. In: WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021, pp. 761–769

Zhang J, Liu B, Tang J, Chen T, Li J (2013) Social influence locality for modeling retweeting behaviors. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence. IJCAI '13, pp. 2761–2767. AAAI Press

Authors and Affiliations

Taha Halal[1] · Bogdan Cautis[2] · Benoît Groz[1] · Ruize Gao[3]

✉ Bogdan Cautis
bogdan.cautis@singaporetech.edu.sg

[1] University Paris-Saclay, Orsay, France

[2] Singapore Institute of Technology, Singapore, Singapore

[3] National University of Singapore, Singapore, Singapore