Received: 23 December 2022 | Revised: 27 November 2023 | Accepted: 8 December 2023

DOI: 10.1002/net.22206

RESEARCH ARTICLE

WILEY

Math-based reinforcement learning for the adaptive budgeted influence maximization problem

Edoardo Fadda[1,2] | **Evelina Di Corso**[2] | **Davide Brusco**[2] | **Vlad Stefan Aelenei**[2] | **Alexandru Balan Rares**[2]

[1]Dept. of Mathematical Sciences, Politecnico di Torino, Torino, Italy
[2]ISIRES Istituto Italiano Ricerca e Sviluppo, Torino, Italy

Correspondence
Edoardo Fadda, Dept. of Mathematical Sciences, Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino 10129, Italy.
Email: edoardo.fadda@polito.it

Abstract

In social networks, the influence maximization problem requires selecting an initial set of nodes to influence so that the spread of influence can reach its maximum under certain diffusion models. Usually, the problem is formulated in a two-stage un-budgeted fashion: The decision maker selects a given number of nodes to influence and observes the results. In the adaptive version of the problem, it is possible to select the nodes at each time step of a given time interval. This allows the decision-maker to exploit the observation of the propagation and to make better decisions. This paper considers the adaptive budgeted influence maximization problem, that is, the adaptive problem in which the decision maker has a finite budget to influence the nodes, and each node requires a cost to be influenced. We present two solution techniques: The first is an approximated value iteration leveraging mixed integer linear problems while the second exploits new concepts from graph neural networks. Extensive numerical experiments demonstrate the effectiveness of the proposed approaches.

KEYWORDS

adaptive budgeted influence maximization problem, approximate dynamic programming, graph neural networks, mixed integer linear problem, reinforcement learning, two-stage stochastic problem

1 | INTRODUCTION

Due to the diffusion of smartphones and tablets, social networks play an important role in our society. According to some sources, around 60% of the world's population has the possibility to use social networks. In Italy, about 58% of the population uses social networks with an average daily usage time of 2 h [9]. Due to these statistics, it is clear that social networks represent a fast and effective way to contact a large audience with a limited effort. This pervasive use of social networks has made them a prime target for advertising. In particular, it has been estimated that 97% of small businesses use social networks to attract new customers [46]. As a result, advertisements are now commonplace across various social networking services, for example, Facebook, Instagram, YouTube, Pinterest, and Twitter.

In the vast area of online marketing, we are interested in *viral marketing*, that is, a business strategy that uses existing social networks to promote content. The name comes from the similarities between how consumers spread information and how viruses infect the population. In order to effectively plan such viral advertising campaigns, it is crucial to understand which individual to target and to analyze both the social network and the influence process. In response to this need, *social computing* emerged as

a field of research studying the computational aspects of social networks [54]. In this wide area, the sub-field of *social influence* studies the behavior of individuals and the spread of information in social networks. It involves interdisciplinary fields such as mathematics, sociology, psychology, computer science, economics, and public safety.

In this sub-field, we address the *influence maximization problem* (IMP), whose aim is to maximize the spread of information in a social network by carefully selecting a subset of users called *seed nodes* [69]. These users are convinced, often through monetary incentives or coupon distribution, to disseminate the information to all their direct connections, such as followers or friends. By doing so, the initial spread of information triggers a propagation process in which every individual forwards (the previously received) information to other people. The field of application of IMP usually encompasses companies willing to promote new products (or services) by using a limited amount of resources (in the form of discounts, free trials, free products, etc.), but it has also applications in other areas such as mental cognition [39], and health [26].

The IMP is an NP-Hard problem [29] and it is difficult to solve even for relatively small networks [69]. Moreover, the problem is usually faced in a single-stage framework in which the set of *seed nodes* is chosen, the propagation of the information is observed and no recourse action is taken. This does not allow the *seed nodes* to be selected in an adaptive manner after observing part of the *diffusion process*. In this paper, we relax this assumption by considering a decision maker that, given a fixed economic budget, wants to maximize the expected number of nodes to influence in a given time period (the duration of the marketing campaign). During each time step within the designated period, she determines the set of nodes to influence, pays the associated cost, and observes the evolution of the diffusion process. We refer to this problem as *adaptive budgeted influence maximization problem* (ABIMP), where we use the term *adaptive* to denote the dynamic structure of the problem (according to [61] and [63]), while in some papers the same dynamic structure is called *online adaptive* [25]. Moreover, we use the term *budgeted* meaning that we consider both monetary budget and influence costs according with the recent literature (see [11, 19, 22, 47]).

The ABIMP is a stochastic multi-stage problem aiming to maximize the expected number of nodes influenced. Traditionally, dynamic programming or multistage stochastic optimization techniques are employed to address this kind of problem. If used in an exact fashion, both methods require the computation of a complex integral, whose probability distribution is not easy to define (being generated by the diffusion process of the information). In addition, the ABIMP contains the IMP as a special case, thus proving to be an NP-Hard problem. Therefore, in order to face the ABIMP, we propose two heuristics: One based on *approximate value iteration (AVI)* and leveraging Integer Linear Problem (ILP) and one using recent advances in Graph Neural Networks (GNNs).

The first approach presents a useful way to deal with dynamic problems involving binary decision variables by leveraging the property of polynomial functions of being easily linearizable.

The second approach employs GNNs to determine the set of nodes to be chosen at each time step. Although *deep neural networks* have been widely used in the field of *reinforcement learning* (RL), leading to the development of *deep reinforcement learning* techniques, GNNs have received less attention despite their capacity to directly handle graph-structured data [2]. By leveraging their characteristics, it becomes feasible to enhance the learning process even in cases where the state space is large. This makes GNNs particularly advantageous in real social network graphs, where efficient learning can significantly enhance performance.

Both approaches are based on concepts from *approximate dynamic programming* or *reinforcement learning*, with a specific focus on using *model-free* approaches (i.e., not assuming a prior knowledge of the diffusion process). This aspect proves to be highly advantageous when deploying these techniques in real-world scenarios, where the diffusion process is not known. Moreover, it is important to highlight that while the proposed heuristics are specifically designed for the ABIMP, the general framework can be applied to other problems exhibiting similar characteristics.

In conclusion, we highlight that the contributions of the paper are:

- Formalize the mathematical model for the ABIMP. To the authors' knowledge, this is a lack in the literature which prevents an easy comparison of the different papers' settings.
- Provide an open-source framework[1] for the ABIMP.
- Propose two heuristics for the ABIMP.
- Provide a rich set of experiments investigating the performance of the two heuristics.

The paper is organized as follows. In Section 2, we review the literature about the problem. In Sections 3 and 4, we present the mathematical formulation of the problem and the proposed solution methods, respectively. Lastly, in Section 5, we show the computational experiments, and in Section 6, we present the conclusion of the work and outline possible future lines of research.

[1] Available at https://github.com/evelina18/adaptive-budgeted-influence-maximization-problem.

2 | LITERATURE REVIEW

In the last decades, the study of the IMP has gained significant attention due to its promising applications in marketing and the growing importance of social networks. Consequently, conducting a comprehensive literature review on this topic is impractical due to the sheer volume of papers available (we refer to [69] for a general review of the topic). For this reason, we concentrate on papers that address applications closely related to the problem at hand. Specifically, we focus on works that tackle adaptive settings and/or consider budget constraints. We examine the solution methods and unique aspects of the problems addressed in those papers. Additionally, we pay special attention to the diffusion process considered. Here, it is important to point out that while the majority of the existing literature focuses on the Independent Cascade Model (ICM) [28] due to its theoretical properties, our primary interest lies in model-free approaches.

The first study considering a two-stage formulation for the IMP is [68]. In their paper, the authors consider a sub-modular second-stage function and obtain cuts applicable in a Benders' fashion. Instead, the first study using ILP to face the IMP is [20]. There, the authors consider an ICM, they model the IMP as a stochastic two-stage maximum covering location problem with uncertain covering sets, and solve it by using a branch-and-cut algorithm based on Benders' decomposition. The same ILP model has been extended in [19] to incorporate budget constraints, resulting in the budgeted IMP (BIMP). In that paper, the author shows that sample average approximation is more efficient than greedy strategies which select nodes based on their features. The BIMP has also been tackled in [47] and in [22]. In the first paper, the authors propose a heuristic based on belief propagation and they focus on ICM. Instead, in the second paper, the authors present two greedy algorithms that select the seed nodes based on the size of the neighborhood, and on the ratio between the cost of the node and the size of the neighborhood, respectively. Additionally, they propose a third approach that combines the outcomes of the two previous ones using simulated annealing. The authors demonstrate that this latter method surpasses previous state-of-the-art heuristics. Finally, two important papers dealing with the multi-period BIMP are [49, 50], which study the influence maximization problem minimizing the total weight of the nodes activated in the first time period.

In more recent years, neural networks have been used to tackle both the IMP and the BIMP. This branch of the literature is beyond the scope of the paper, but it is important to highlight that recently, in [31] the authors use GNN to tackle the IMP. Despite being applied in another setting, this heuristic can be seen as the ancestor of the GNN-based heuristic that we will present.

All the aforementioned papers consider static seeding selection, that is, the initial seed is determined and the outcome is observed without any recourse action. This makes these approaches less effective in a dynamic setting, where the decision maker can adapt the choice of the seed set to the realization of the diffusion process [63]. Despite the wide interest in two-stage formulations, only a minor part of the literature focuses on the adaptive influence maximization problem (AIMP).

To our knowledge, the first paper presenting AIMP is [55]. There, the authors develop a concave approximation of the objective function and use hill climbing to optimize it. Their contribution is mostly related to the theoretical bound in the case of ICM. Instead, in [63] the authors propose an adaptive seeding strategy based on a greedy strategy which selects in each time step the node that is able to maximize the increment in the objective function. By means of computational examples, they show the effectiveness of the proposed method considering ICMs.

In more recent years, model-free RL techniques addressing the AIMP have gained traction. The key advantage of employing these techniques is that they do not require any assumption on the diffusion process and they can adapt and learn optimal strategies directly from data, making them highly flexible and effective. In this branch, in [65] the authors face the problem via graph embedding: Each node of the graph is represented through a set of features and a RL model is formulated where both the states and the actions are represented in a lower dimensional space. Another important contribution leveraging RL is presented in [40], where the authors combine evolutionary algorithms with deep RL algorithms for solving the AIMP in complex networks. The mix of these two techniques enables the proposed meta-heuristic to obtain a good trade-off between effectiveness and efficiency. A similar technique has been used in the context of adaptive topic aware influence maximization in [60], where the authors apply Double Deep Q-Networks with prioritization of experience replay, to train larger models.

It is worth noting that the literature about AIMP usually considers a given amount of nodes to be selected in each time step of the time horizon. For example, in [16, 17, 40, 63–65] the authors present approaches where one node is selected per time instant while in [24], and [21] the authors present methodologies where a constant quantity of node per time instant is selected. Whether the first assumption may lead to problems with a time constraint [62], the second one is restrictive. For this reason, in the proposed methods we remove this assumption letting the decision maker to select the number of nodes that she considers more appropriate in each time step.

Despite AIMP being a special case of ABIMP, where the costs for influencing nodes are all equal to one and the budget is equal to the seed size, the existing literature primarily focuses on this special setting, with only a limited number of papers addressing the more general one. To our knowledge, the only papers considering budget constraints in the adaptive setting are: [11], [66], and [70]. All of them employ the ICM only. The first paper explores a slightly different problem, where there are

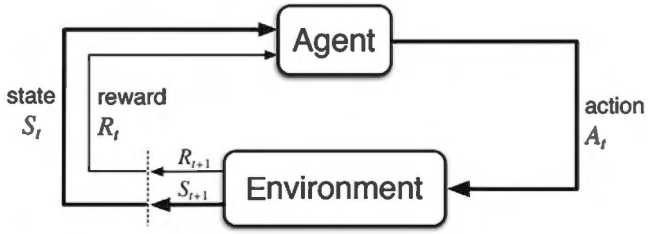

FIGURE 1 Agent–environment interaction in a MDP [58].

different diffusion processes (each one associated with a feature of the product), and each node is characterized by a purchasing probability. The authors develop a greedy algorithm that selects the nodes leading to the greatest marginal contribution. They consider networks with up to approximately 75.9k nodes and 508.8k arcs, enabling the examination of larger-scale scenarios [11]. The second paper introduces an optimization algorithm that learns the parameters of the ICM over time. The authors compare various techniques and prove that their proposed learning strategy surpasses baseline approaches, achieving rewards that closely resemble those obtained by a greedy policy with perfect knowledge of the diffusion parameters. The experiments are conducted on networks consisting of less than 300 nodes and 6000 arcs [66]. Lastly, in the third paper, the authors develop a greedy heuristic selecting in each time step the nodes leading to the greatest immediate number of influenced nodes. The study deals with graphs of 15.2k nodes and 58.9k arcs, providing valuable insights into the effectiveness of adaptive strategies in practical scenarios [70].

In comparison to the aforementioned studies, the heuristics proposed in this paper can handle larger graphs and broaden the method's applicability as they do not rely on assumptions about the diffusion model.

3 | PROBLEM STATEMENT

Let us consider a marketing campaign lasting for a finite set of time steps $\mathcal{T} = \{0, \dots, T\}$. We model the social network as a directed graph $G = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V} = \{1, \dots, V\}$ is the set of nodes representing the individuals and $\mathcal{A} \subset \mathcal{V} \times \mathcal{V}$ is the set of arcs representing the relationship among them. Each node $v \in \mathcal{V}$ is characterized by:

- $s_v^t \in \{0, 1\}$: The state of the node, equal to 1 if it is *activated* (or *influenced*) at time t or equal to zero otherwise.
- $c_v \in \mathbb{R}^+$: The cost to pay for directly influencing node v.
- $\mathcal{N}^{out}(v) = \{u \text{ s.t. } (v, u) \in \mathcal{A}\}$: The set of nodes that can be directly influenced by node v.
- $\mathcal{N}^{in}(v) = \{u \text{ s.t. } (u, v) \in \mathcal{A}\}$: The set of nodes that may directly influence the state of node v.

The set $\mathcal{N}^{in}(v)$ influences the state of node v through

$$s_v^{t+1} = f(s_u^t, u \in \mathcal{N}^{in}(v), \xi_v^t), \tag{1}$$

where the function $f(\cdot)$ describes the diffusion process, and ξ_v^t is a random parameter modeling the stochastic nature of the process. We consider two types of models, namely the *generalized threshold model* and the *complex threshold model*. While the first one has been proposed in [28], we define the second one in order to better address the peculiarities of online social networks. We describe these models in Subsections 3.1 and 3.2, respectively.

In each time step $t \in \mathcal{T}$, the set of nodes can be divided into two disjoint sets: $\mathcal{V} = \mathcal{V}_t^A \cup \mathcal{V}_t^I$, where: $\mathcal{V}_t^A = \{v \in \mathcal{V} \text{ s.t. } s_v^t = 1\}$ is the set of active nodes and $\mathcal{V}_t^I = \{v \in \mathcal{V} \text{ s.t. } s_v^t = 0\}$ is the set of inactive nodes. We call the number of influenced nodes at time t, $I_t = |\mathcal{V}_t^A|$, that is, the cardinality of the set $\mathcal{V}_t^A$.

It is important to notice that the state of each node at $t+1$ is only influenced by the state of the nodes at time t (see Equation 1), thus the dynamic of the problem satisfies the Markov property and it can be described as a *Markov Decision Process* (MDP) [58]. MDPs are meant to be a framing of the problem of learning from interaction to achieve a specific objective. In this context, the decision maker is called *agent*, while the entity it interacts with is termed *environment*. The interaction between the agent and the environment is continuous and dynamic. The agent chooses *actions* (in the ABIMP the set of nodes to select) and the environment responds to these actions by presenting new situations (called *states*) and giving *rewards* (i.e., special numerical values that the agent aims to maximize over time by making optimal choices of actions). A graphical representation of this interaction is shown in Figure 1.

In the ABIMP, the agent has an initial budget, called B, that is used to influence nodes throughout the marketing campaign. During each time step $t \in \mathcal{T}$, the residual budget b_t defines the set of feasible actions

$$\mathcal{X}_t = \left\{ x_v^t \in \{0, 1\} \; \forall \, v \in \mathcal{V}, \text{ s.t. } \sum_{v \in \mathcal{V}} c_v x_v^t \leq b_t \right\} \qquad \forall \, t = 0, \dots, T, \tag{2}$$

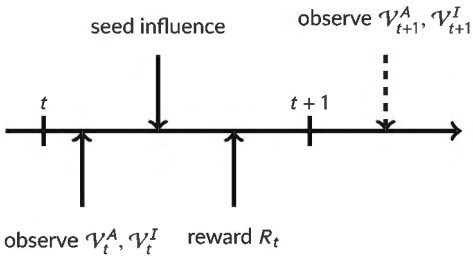

FIGURE 2 Flow of information.

where x_v^t are the decision variables equal to 1 if the agent decides to select node v at time t, and 0 otherwise. We call $x_t \in \{0,1\}^V$ the vector whose components are x_v^t, $\forall\, v \in \mathcal{V}$. It is worth noticing that in Equation (2), we may select all the nodes and not just the ones in $\mathcal{V}_t^I$ since, in principle, it is possible to select several times a node in $\mathcal{V}_t^A$ to trigger again the diffusion process.

The state of the system is composed of the state of each node and the available budget, that is,

$$S_t = (s_1^t, \dots, s_V^t, b_t). \tag{3}$$

Given the state S_t and the action x_v^t $\forall\, v \in \mathcal{V}$, the next state is

$$S_{t+1} = \begin{bmatrix} f(\min[1, s_u^t + x_u^t], u \in \mathcal{N}^{in}(1), \xi_1^t) \\ \dots \\ f(\min[1, s_u^t + x_u^t], u \in \mathcal{N}^{in}(V), \xi_V^t) \\ b_t - \sum_{v\in\mathcal{V}} c_v x_v^t \end{bmatrix}. \tag{4}$$

Finally, the reward that the agent receives is

$$R_t(S_t, x_t) = \begin{cases} 0 & \text{for } t \neq T, \\ \lambda I_T + (b_T - \sum_{v\in\mathcal{V}} c_v x_v^T) & \text{for } t = T, \end{cases} \tag{5}$$

where λ is a coefficient that quantifies the economic value of each influenced node (it can be computed, e.g., by the expected purchases given the current number of active nodes). Equation (5) imposes that the only reward is at $t = T$, and no reward is given for $t \neq T$. In such scenarios, the agent often encounters long sequences of actions and states with no immediate rewards, making it challenging to learn an effective policy or strategy because no feedback is received. This characteristic is known as reward sparseness [53]. To mitigate this issue, according to [37, 41, 60], we consider as reward the marginal contribution of each time step. In formula:

$$R_t(S_t, x_t) = \lambda(I_t - I_{t-1}) - \sum_{v\in\mathcal{V}} c_v x_v^t \qquad \forall\, t = 0, \dots, T, \tag{6}$$

where we set $I_{-1} = I_0 = 0$ since we assume that no node is active without our intervention. It is important to notice that Equations (5) and (6) are really close to each other. In fact, from Equation (5), we have that:

$$\sum_{t=0}^{T} R_t(S_t, x_t) = \lambda I_T + \left(b_T - \sum_{v\in\mathcal{V}} c_v x_v^T\right) = \lambda I_T + B - \sum_{t=0}^{T}\sum_{v\in\mathcal{V}} c_v x_v^t, \tag{7}$$

where the last equality holds since $b_T = b_{T-1} - \sum_{v\in\mathcal{V}} c_v x_v^{T-1} = b_{T-2} - \sum_{v\in\mathcal{V}} c_v x_v^{T-2} - \sum_{v\in\mathcal{V}} c_v x_v^{T-1} = \dots = B - \sum_{t=0}^{T-1}\sum_{v\in\mathcal{V}} c_v x_v^t$. Instead, summing over the time horizon the reward of Equation (6), we get:

$$\begin{aligned} \sum_{t=0}^{T} R_t(S_t, x_t) &= \sum_{t=0}^{T}\left(\lambda(I_t - I_{t-1}) - \sum_{v\in\mathcal{V}} c_v x_v^t\right) = \lambda\sum_{t=0}^{T}(I_t - I_{t-1}) - \sum_{t=0}^{T}\sum_{v\in\mathcal{V}} c_v x_v^t \\ &= \lambda I_T - \sum_{t=0}^{T}\sum_{v\in\mathcal{V}} c_v x_v^t. \end{aligned} \tag{8}$$

Equations (7) and (8) differ from a constant term, which does not affect the solutions of the underlying optimization problem. Moreover, it is important to notice that $R_t(S_t, x_t)$ is deterministic since the effect of x_t manifests in $t + 1$. In other words, when we decide whose set of nodes to select, both I_t and I_{t-1} are known (Figure 2 depicts the flow of information).

The goal of the decision-maker is to maximize the expected sum of the rewards expressed as:

$$\mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R_t(S_t, x_t)\right], \tag{9}$$

where γ is a discount factor that weighs present and future rewards, encouraging the agent to prioritize short-term gains over long-term ones [7]. In this setting, it is possible to define a *value function* $V_t(S_t)$ that quantifies the expected reward obtained when applying an optimal sequence of action from time t on-wards, starting from state S_t. This function can be computed using the Bellman equation:

$$V_t(S_t) = \max_{x \in \mathcal{X}_t} \{R_t(S_t, x) + \gamma \mathbb{E}[V_{t+1}(S_{t+1}) | S_t, x]\} \qquad \forall\, t = 0, \ldots, T-1. \tag{10}$$

In order to compute Equation (10), it is necessary to solve a difficult stochastic optimization problem. Nevertheless, it is possible to reformulate the problem and switch optimization and expectation employing post-decision state variables. These variables describe the state of the system after the decision has been taken but before the noise realization. In the ABIMP, they can be expressed as

$$S^x_{t+1} = \begin{bmatrix} \min[1, s^t_1 + x^t_1] \\ \cdots \\ \min[1, s^t_V + x^t_V] \\ b_t - \sum_{v \in \mathcal{V}} c_v x^t_v \end{bmatrix}. \tag{11}$$

Using Equation (11), we define the value function around post-decision states as

$$V^x_t(S^x_t) = \mathbb{E}\left[V_{t+1}(S_{t+1}) | S^x_t\right]. \tag{12}$$

Plugging Equations (12) in (10), and noting that $\mathbb{E}[\cdot | S_t, x] = \mathbb{E}[\cdot | S^x_t]$, we obtain:

$$V_t(S_t) = \max_{x \in \mathcal{X}_t} \{R(S_t, x) + \gamma V^x_t(S^x_t)\}. \tag{13}$$

Taking the expectation at $t-1$, we get the Bellman equation for the post-decision value function:

$$V^x_{t-1}\left(S^x_{t-1}\right) = \mathbb{E}\left[V_t(S_t) | S^x_{t-1}\right] = \mathbb{E}\left[\max_{x \in \mathcal{X}} \{R(S_t, x) + \gamma V^x_t(S^x_t)\}\right]. \tag{14}$$

Equation (14) allows swapping optimization and expectations with respect to Equation (10). Therefore, Equation (14) requires the solution of several deterministic optimization problems instead of one solution of a stochastic one (as Equation 10). Moreover, the expectation in Equation (14) can be tackled through statistical learning techniques.

It can be noticed that dynamic programming recursion in terms of Q-factors (see, e.g., [58]) is close to Equation (14), since the state–action pairs may be regarded as post-decision states. Nevertheless, when they can be applied, post-decision states are more convenient than Q-factors since they require a space of the same dimension as the initial state space.

3.1 | General threshold model

The *general threshold model* introduced by [29] and [45], is one of the most commonly used diffusion models. It defines Equation (1) as:

$$s^{t+1}_v = \begin{cases} 1 & \text{if } \phi(s^t_u,\ u \in \mathcal{N}^{in}(v)) \geq \theta_v \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where $\theta_v \in [0, 1]$ is a threshold values and $\phi : \{s^t_u,\ u \in \mathcal{N}^{in}(v)\} \to [0, 1]$ is called *activation function*. It is assumed to be monotonic, reflecting that the more active nodes in $\mathcal{N}^{in}(v)$, the greater the likelihood of node v of being activated. By fixing the activation function, we get different types of models. In particular, we consider the *linear threshold model* (LTM) and the *independent cascade model* (ICM) [28].

In the LTM, each arc of the network is associated with a weight p_{uv} (such that $\sum_{u \in \mathcal{N}^{in}(v)} p_{uv} \leq 1$), and each node v has a threshold $\theta_v \in [0, 1]$. In this setting, Equation (15) becomes

$$s^{t+1}_v = \begin{cases} 1 & \text{if } \sum_{u \in \mathcal{N}^{in}(v)} p_{uv} s^t_u \geq \theta_v \\ 0 & \text{otherwise} \end{cases}. \tag{16}$$

Therefore, in the LTM, a node v becomes active if a weighted sum of the active nodes in the neighborhood is greater than its resistance θ_v. We consider the θ_v to be uniformly distributed in $[0, 1]$, and that $\sum_{u \in \mathcal{N}^{in}(v)} p_{uv} = 1\ \forall v \in \mathcal{V}$.

In the ICM, each arc of the network is associated with the probability that node u influences node v if u is active, we call it p_{uv}. In this setting, Equation (15) becomes

$$s^{t+1}_v = \begin{cases} 1 & \text{if } \sum_{u \in \mathcal{N}^{in}(v)} X_{uv} s^t_u \geq 1, \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where X_{uv} are Bernoulli random variables with probability p_{uv}. In this model, node v becomes active if it is influenced by at least one active node in $\mathcal{N}^{in}(v)$, and each one of them has a probability p_{uv} of doing so. It is important to notice that, once that node u becomes active, the realization of the X_{uv} is fixed. In other words, if node u becomes active at time t and $X_{uv} = 0$, then node u will not have other opportunities to influence node v in the future. Hence, in this diffusion model, there is no point in influencing twice the same node.

3.2 | Complex threshold model

Based on the general threshold model, we define the *Complex Threshold Model* (CTM) by considering Equation (1) to be

$$s_v^t = \begin{cases} 1 & \exists\, u \in \mathcal{N}^{in}(v) \cap \mathcal{V}_t^I \text{ s.t. } \mathrm{P}_t(u) \geq \mathrm{R}(v, \xi_t), \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

where $\mathrm{P}_t(u)$ represents the *influence power* of node u at time t, while $\mathrm{R}(v, \xi_t)$ represents the random *influence resistance* of node v. Therefore, a node becomes active if there exists at least one of the neighborhood nodes which has an influence greater than its resistance. As in [67], we divide the nodes in *influencer*, *super-active*, *active*, or *passive*, and we call $\mathcal{K}$ the set of node types (it defines a partition on $\mathcal{V}$).

We assume that the influence resistances are distributed according to a beta distribution ($\mathrm{B}(\alpha_k, \beta_k)$, where α_k, β_k depend on the type of node), and summed to a noise $\xi_t \sim \mathcal{U}[-0.05, +0.05]$.

In contrast with the ICM, each node tries to influence all its neighborhoods during each time step. This enables the model to account for content re-share and allows the agent to select several times the same node. Nevertheless, the content that is re-shared does not have the same influence power as the first one. This is modelled through the influence power

$$\mathrm{P}_{t+1}(v) = \mathrm{P}_0(v) \prod_{\tau=0}^{t} (1 - \zeta_k s_v^\tau), \quad \forall\, k \in \mathcal{K} \tag{19}$$

where ζ_k a rate of decay that depends on the type of node, and

$$\mathrm{P}_0(v) = 1 - e^{-\lambda_k |\mathcal{N}^{out}(v)| / \max_v |\mathcal{N}^{out}(v)|}, \quad \forall\, k \in \mathcal{K}, \tag{20}$$

where λ_k, is distributed according to a uniform distribution whose lower and upper bound depend on the type of node k. It is worth noting that in order to have a Markov representation for this model, we should also collect the values of $\mathrm{P}_t(v)$ as part of the state. Nevertheless, we decided to stick with the state definition of Equation (3) in order to test the ability of the proposed techniques to deal with difficult diffusion processes.

4 | SOLUTION METHODS

In this section, we present the proposed solution methods for the ABIMP. In particular, in Subsection 4.1, we present a customized version of the model described in [20] that will be used as a benchmark; in Subsection 4.2 we describe the heuristic based on reinforcement learning and, in Subsection 4.3 we present the heuristic based on graph neural network.

4.1 | Two-stage budgeted influence maximization problem

A two-stage mathematical programming formulation for the BIMP for an ICM has been proposed in [19]. Being a two-stage stochastic optimization problem, we define the set of scenarios $\mathcal{S}$. Each scenario $s \in \mathcal{S}$ represents a possible realization of the diffusion process (e.g., a possible realization of all the Bernoulli random variables described in Equation 17) and has a realization probability π_s. Given a scenario s, we call $G^s = (\mathcal{V}, \mathcal{A}_s)$, the graph whose set of arcs $\mathcal{A}_s$ contains only those for which the realization of X_{uv} is equal to 1 in scenario s. Following [19], we define the *reachability set* (also called *random reverse reachable* set in [6, 59]) $\mathcal{R}(s, v)$ as the set of nodes that influence node v in scenario s. In other words, $\mathcal{R}(s, v)$ is the set of all the nodes from which there is a directed path to v in G^s. Therefore, node u influence node v in scenario s if $u \in \mathcal{R}(s, v)$. For each scenario and node, sets $\mathcal{R}(s, v)$ can be easily determined by, for example, a reverse breadth-first search starting from v in graph G^s.

The decision variables of the model are:

- x_v which assumes value 1 if the node v is selected in the seed set;
- y_v^s which assumes value 1 if node v is influenced in scenario s.

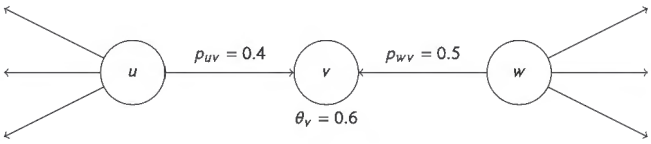

FIGURE 3 Subgraph showing that the superposition principle is not valid in a LTM model.

The mathematical model is:

$$\max \quad \lambda \sum_{s \in S} \pi_s \sum_{v \in \mathcal{V}} y_v^s - \sum_{v \in V} c_v x_v \tag{21}$$

$$\text{s.t.} \quad \sum_{v \in V} c_v x_v \leq B \tag{22}$$

$$y_v^s \leq \sum_{u \in \mathcal{R}(s,v)} x_u \qquad \forall v \in \mathcal{V}, \forall s \in S \tag{23}$$

$$x_v \in \{0,1\} \qquad \forall v \in \mathcal{V} \tag{24}$$

$$y_v^s \in \{0,1\} \qquad \forall v \in \mathcal{V}, \forall s \in S. \tag{25}$$

The objective function (21) maximizes the expected profit coming from the influenced nodes minus the cost for influencing (it is equivalent, up to the constant B to Equation 5). Constraint (22) limits the budget that can be used to select the initial seed and Constraints (23) allow a node to be influenced only if a node in the reachability set has been activated. Finally, Constraints (24) and (25) force all the decision variables to be binary.

As noticed in [20], the variables y_v^s can be relaxed to be continuous, thus it is possible to solve the problem using *Benders' decomposition*. Moreover, all the results obtained in [20], for the unbudgeted case, still hold since the only difference with respect to their model is in Constraint (22) and it only affects the master problem while all the sub-problems remain the same. For these reasons, in the computational experiments, we apply the solution method proposed in [20] to get the solution of Model (21)–(25).

It is worth noting that this approach can be seen as an alternative to the *Reverse Influence Sampling* proposed in [6], where the stochastic maximum coverage problem is solved with a mathematical model and not via a heuristic algorithm. Model (21)–(25) can deal with all the diffusion processes satisfying a *superposition principle* since Constraints (23) are linear. This may not be the case in several diffusion processes in which the effect of influencing two nodes can lead to a diffusion much greater than the superposition of the two processes. To see this, consider the graph in Figure 3 and an LTM. Neither node u, nor node w are in the reachable space of node v since $p_{uv} = 0.4 < \theta_v = 0.6$, and $p_{wv} = 0.5 < \theta_v = 0.6$. Nevertheless, if both nodes are influenced, it is possible to influence v ($p_{uv} + p_{wv} = 0.9 \geq \theta_v = 0.6$).

Despite this lack, Model (21)–(25) represents an interesting benchmark, and it can be solved for small instances by using exact solvers. In the following, we refer to the two-stage formulation approach as `TS`.

4.2 | Approximate value iteration

In this section, we propose a new solution method based on the approximation of the post-decision value function, defined in Equation (12). The general pseudo-code of the method is shown in Algorithm 1, (see, e.g., [7] p. 183, or [48] p. 128). The algorithm starts from an initial approximation, called $\hat{V}_t^x$. Then, for a given number of times (called `n_episodes`), it runs an episode lasting T time steps starting from the initial state S_0. During time step t the action maximizing the sum of the actual reward and the discounted future reward (estimated by $\hat{V}_t^x$) is computed and S_t^x, $R(S_t, x)$, $v_t^x = \max_{x \in \mathcal{X}_t} R(S_t, x) + \gamma \hat{V}_t^x[S_t^x]$ are used to update $\hat{V}_{t-1}^x$ (if $t > 0$). Calling x_t^* the optimal solution the agent applies x_t^* with probability $1 - \varepsilon$ or another random action with probability ε. Finally, the new state S_{t+1} is reached.

Algorithm 1 is the general learning framework called approximate value iteration which is the core of several reinforcement learning algorithms. To effectively apply it to the ABIMP we have to specify the form of the post-decision value function approximation, and how to learn from the information gathered from the episodes. Problem (26) can be formalized by considering the definition of the feasible set in Equation (2), and the immediate reward in Equation (6). In detail, avoiding index t for notation convenience, Problem (26) can be written as:

Algorithm 1. Approximate value iteration

Initialize $\hat{V}_t^x[\cdot], \forall t = 0 : T-1, \hat{V}_T^x[\cdot] = b_T - \sum_v c_v x_v$.
for i = 1, …, `n_episodes` **do**
  $S_0 = (0, \dots, 0, B)$. ▷ The last component is the initial budget.
  Generate a random realization of the noises ξ_v^t, $\forall\, v \in \mathcal{V}$, $t = 0 : T+1$.
  for $t = 0 : T$ **do**
    Solve:

$$v_t^x = \max_{x \in \mathcal{X}_t} R(S_t, x) + \gamma \hat{V}_t^x[S_t^x]. \tag{26}$$

    Store S_t^x, $R(S_t, x)$, v_t^x
    Let x_t^* be the optimal solution of problem (26).
    Use a ε-greedy policy which applies x_t^* or a random action.
    Compute the next pre-decision state S_{t+1} using S_t^x and ξ_v^{t+1}.
    if t % N == 0 **then**
      Use the data stored to update $\hat{V}_{t-1}^x$.
    end if
  end for
end for

$$\max \quad \sum_{v \in \mathcal{V}} \left(-c_v x_v + \gamma \hat{V}^x[s_v + x_v] \right) \tag{27}$$

$$\text{s.t.} \quad \sum_{v \in \mathcal{V}} c_v x_v \le b \tag{28}$$

$$x_v \in \{0, 1\} \qquad \forall\, v \in \mathcal{V}. \tag{29}$$

The objective function (27) maximizes the sum of the immediate reward plus an estimation of the expected future reward. Equation (27) follows from Equation (6) where, since the contribution $\lambda(I_t - I_{t-1})$ is defined by x_{t-1} and S_{t-1}, it is a constant thus it is removed. Constraint (28) limits the budget that can be used, and Constraints (29) impose that the x_v must be binary variables. Model (27)–(29) has a knapsack structure. Since large instances of the knapsack problem with linear objective functions can be solved in a reasonable time by exact solvers, we may resort to a linear approximation of $\hat{V}^x[\cdot]_t$, that is,

$$\hat{V}_t^x[z] = \sum_{v \in \mathcal{V}} \omega_v^t z_v, \quad \forall\, t = 1, \dots, T-1 \tag{30}$$

where ω_v^t is a parameter $\forall t, v$. This would require the estimation of $V \cdot T$ coefficients, which becomes impractical for large instances. Thus, we define a set I of features that can be computed from the node characteristics (we call them $f_i^{t,v}$), and a set of parameters a_i equal for all the nodes and for every time step. In formula,

$$\omega_v^t = \sum_{i=1}^{I} f_i^{t,v} a_i \qquad \forall v \in \mathcal{V}, \quad \forall\, t = 1, \dots, T-1. \tag{31}$$

Using Equation (31), we have to calibrate I parameters (the a_i) instead of $V \cdot T$ parameters. Nevertheless, as pointed out in Section 4.1, linear functions are not able to describe the joint effect of influencing more nodes at the same time. For example, using linear function it is impossible to model when two nodes share a large fraction of their neighborhood. Therefore, we use as $\hat{V}_t^x[\cdot]$ a set of polynomial functions (one for each t). Interestingly, polynomial functions can be linearized since variables x_v are binary [4]. In fact, we can write the objective function (27), considering index t, as

$$\max \sum_{v \in \mathcal{V}} - c_v x_v^t + \sum_{A \in 2^{|\mathcal{V}|}} w_A^t \prod_{v \in A} x_v^t, \tag{32}$$

where w_A^t is the parameter that weighs the interactions obtained by influencing all the nodes in the subset A at time t. The product $\prod_{v \in A} x_v^t$ can be linearized, by introducing a variable z_A^t, satisfying the following constraints:

$$\sum_{v \in A} x_v^t \le |A| - 1 + z_A^t \qquad \text{if } w_A^t < 0$$

$$x_v^t \ge z_A^t \qquad \forall\, v \in A \quad \text{if } w_A^t > 0. \tag{33}$$

Clearly, Equation (33) is used for all the sets A with $|A| \geq 2$, while for the sets containing just one element the product in (32) is a single variable.

To estimate the parameters w_A^t, we generalize the set I of features and the corresponding parameters a_i to higher order by defining the sets $I^{(1)}, I^{(2)}, \dots, I^{(V)}$ of features of degree $1, 2, \dots, V$ and the corresponding parameters $a_i^{(1)}, a_i^{(2)}, \dots, a_i^{(V)}$ (with this choice we rename a_i as $a_i^{(1)}$ and I as $I^{(1)}$ to obtain a uniform notation). Thus, given a subset of nodes A the corresponding coefficient w_A^t in Equation (32) is defined as

$$w_A^t = \sum_{i=1}^{I^{(|A|)}} f_i^{t,A} a_i^{I^{(|A|)}} \qquad \forall\, t = 1, \dots, T-1, \quad \forall A \in 2^V. \tag{34}$$

Notice that while the features $f_i^{t,A}$ depends by the set A and by the time step, the parameters $a_i^{I^{(|A|)}}$ just depend by the feature $i \in I^{(|A|)}$. Therefore, the final expression of the post-decision value function approximation is

$$\hat{V}_t^x[z] = \sum_{A \in 2^{\mathcal{V}}} w_A^t \prod_{v \in A} z_v = \sum_{A \in 2^{\mathcal{V}}} \sum_{i=1}^{I^{(|A|)}} f_i^{t,A} a_i^{I^{(|A|)}} \prod_{v \in A} z_v, \tag{35}$$

where the parameters to estimate are $a_i^{(1)}, a_i^{(2)}, \dots, a_i^{(V)}$. Considering all the sets $A \in 2^{\mathcal{V}}$ makes Model (27)-(29) impossible to solve even for small-size instances. For this reason, we need to define a reasonable way to fit the polynomial estimation of $\hat{V}_t^x[\cdot]$. To achieve this goal, we develop the iterative procedure described in Algorithm 2. First, we select a maximum degree of the polynomial approximation $\hat{V}_t^x[\cdot]$ called M and we solve Model (27)–(29) with linear $\hat{V}_t^x[\cdot]$, that is, the approximation in Equation (30). Then, we consider the set of all nodes for which $x_v = 1$ and, for each possible couple, we add a second-order approximation term. In other words, if $x_u = 1$ and $x_v = 1$, we add in the $\hat{V}_t^x[\cdot]$ a term of the form $w_{uv}^t x_u x_v$, where $w_{uv}^t = \sum_{i=1}^{I^{(2)}} f_i^{t,uv} a_i^{(2)}$. By solving this quadratic problem it is possible that the optimal solution changes, for example, if the corresponding w_{uv}^t is negative since node u and v influence sets of nodes having a big intersection. In this case, we repeat the procedure by updating the second-order terms. Otherwise, if the optimal solution remains the same, a higher-order approximation is considered. We repeat these steps until the maximum polynomial degree M is reached.

Algorithm 2. Iterative solution (AVI)

Choose a maximum degree M.
Initialize $\hat{V}_t^x[\cdot]$ of degree 1.
$x \leftarrow$ solution of Model (27)–(29).
$k = 2$
while $k \neq M$ **do**
  collect $x_v = 1$
  $\hat{V}_t^x[\cdot] \leftarrow$ update $\hat{V}_t^x[\cdot]$ by considering the k order terms defined by the $x_v = 1$.
  $x' \leftarrow$ solution of Model (27)–(29).
  if x == x' **then**
    $k \leftarrow k + 1$.
  else
    $x \leftarrow x'$.
  end if
end while

This procedure requires the solutions of several ILPs. Thus, it is of paramount importance to not re-compute the solution from scratches. In fact, it is easy to notice that the differences between the models considered in two subsequent iterations of Algorithm 2 are just terms in the objective function and constraints regarding the polynomial expansion of the value function. Thus, the optimal solution computed in one iteration is feasible for the model considered in the next one and it usually provides a good starting solution. Without this consideration, this algorithm cannot be applied due to the time required for the learning process.

Algorithm 2 is run to solve Problem (26). Then, the algorithm uses S_t^x, $R(S_t, x)$, and v_t^x are collected, and once every N steps a min squared error problem is solved, where the error is computed as

$$\begin{bmatrix} R(S_t,x)+\gamma v_t^x \\ \cdots \\ R(S_{t-(N-1)},x)+\gamma v_{t-(N-1)}^x \end{bmatrix} - \left[\begin{array}{ccc|ccc} \sum_{v:x_v^{t-1}=1} f_1^{t-1,v} & \cdots & \sum_{v:x_v^{t-1}=1} f_{I^{(1)}}^{t-1,v} & \sum_{(u,v):x_u^{t-1}=1,x_v^{t-1}=1} f_1^{t-1,uv} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \sum_{v:x_v^{t-(N-2)}=1} f_1^{t-(N-2),v} & \cdots & \sum_{v:x_v^{t-(N-2)}=1} f_{I^{(1)}}^{t-(N-2),v} & \sum_{(u,v):x_u^{t-(N-2)}=1,x_v^{t-(N-2)}=1} f_1^{t-(N-2),uv} & \cdots \end{array}\right] \begin{bmatrix} a_1^{(1)} \\ \vdots \\ a_{I^{(1)}}^{(1)} \\ \hline a_1^{(2)} \\ \vdots \end{bmatrix}. \tag{36}$$

Calling $a_i^{*(k)}, k = 1, \ldots, M$ the solution of this problem, the coefficients $a_i^{(k)}$ are updated as follows:

$$a_i^{(k)} \leftarrow \alpha a_i^{*(k)} + (1-\alpha) a_i^{(k)}, \quad \forall k = 1, \ldots, M,$$

where α is a given learning rate.

In the following, we refer to the algorithm presented in this section as `AVI` (approximate value iteration).

4.3 | Graph neural network

In this Subsection, we provide a detailed description of the graph neural network approach based on the Double Deep Q-Networks (DDQN) algorithm proposed in [23]. The DDQN algorithm enhances Deep Q-learning by utilizing two separate Q-value estimators instead of one [23]. Doing this, DDQN addresses the problem of overestimation bias that can occur in traditional Q-Learning. The two networks considered are the *online network*, which is used to select actions, and the *target network*, which provides a more accurate evaluation of the selected action's Q-value [33, 57]. In our work, we propose three key improvements to better adapt this framework to the problem at hand.

Firstly, since the ABIMP deals with graphs, we incorporate graph embedding techniques. This involves mapping the nodes, edges, and their respective features into a lower-dimensional vector space, while preserving essential properties such as graph structure and information [18]. By leveraging graph embedding, we significantly enhance the generalization capability of the neural network, enabling it to adapt to varying graph topologies during testing without requiring retraining.

Secondly, instead of focusing on Q-factors, we use post-decision state variables. This approach allows us to use a much more compact representation. In fact, instead of estimating the value for each possible state-action pair (S, x), we only need to estimate the value for each possible state S. This reduction in dimensionality simplifies the learning process and improves computational efficiency.

Thirdly, we adapt the technique to accommodate the available budget constraint. To achieve this, we employ *action masking* (i.e., we represent the subset of feasible actions that the agent can choose from), which limits the budget to be spent at each time step to the available budget b_t. By incorporating action masking, we ensure that the agent operates within the specified budgetary limits.

The general algorithm is reported in Algorithm 3.

In the first step, the two GNNs called Q^O (i.e., online network) and Q^T (i.e., target network) are initialized. Both networks take as input a post-decision state and return an approximation of the post-decision value function. Then, during each time step of each episode, one of the two GNNs is randomly selected, used to get an ε-greedy action, and updated using the value function estimation computed by the other one. During the training phase, the learning is performed every N time steps using a batching strategy. More in detail, in each time step of the procedure, the actual state S_t, action x_t, reward r_t, next state S_{t+1}, budget b_t and time t are collected in a dataset called H. Then, when the learning procedure is run, a random sample of dimension n is selected from H and it is used to update both networks. The process continues until all episodes and time steps are completed.

In the following, we refer to this algorithm as `GNNQL` (graph neural network Q-learning).

5 | COMPUTATIONAL EXPERIMENTS

In this section, we test the performance of the proposed methodologies, namely `AVI` and `GNNQL`.

For both `AVI` and `GNNQL`, we consider the following parameters:

- $\gamma = 0.9$;

Algorithm 3. GNNQL

$H = \emptyset$.
Initialize Q^O and Q^T.
for i = 1, …, n_episodes **do**
  generate a random realization of the noises ξ_v^t, $\forall\, v \in \mathcal{V}$, $t = 0 : T-1$.
  $S_0 = (0, \dots, 0, B)$. ▷ The last component is the initial budget.
  for $t = 0, \dots, T-1$ **do**
    Consider state S_t.
    Q = randomly select Q^O or Q^T.
    Let x_t^* be $max_x Q(S_t + x)$.
    Use a ε-greedy policy which applies x_t^* or a random action.
    Compute the next pre-decision state S_{t+1} and the reward r_t.
    Add (S_t, x_t, r_t, S_{t+1}) in H.
    if $Q == Q^O$ **then**
      $Q^O \longleftarrow (1-\alpha)Q^O + \alpha(r_t + \gamma \max_a Q^T(S_{t+1} + x_t))$.
    else
      $Q^T \longleftarrow (1-\alpha)Q^T + \alpha(r_t + \gamma \max_a Q^O(S_{t+1} + x_t))$.
    end if
    if t % N == 0 **then**
      $\mathcal{B} \longleftarrow$randomly select n samples from H.
      Update Q^O and Q^T with $\mathcal{B}$.
      $H = \emptyset$.
    end if
  end for
end for

- $\alpha = 0.4$;
- $N = T/4$;
- ε is set to 0.5 and it decays according to $\frac{1}{i}$, where i is the number of episodes considered.

For AVI, we use the maximum degree of the post-decision value function approximation to 3 (i.e., $M = 3$). Moreover, we consider the following features:

- First-order features:
  1. the state and the cost of the node;
  2. the number of neighborhood nodes (both $|\mathcal{N}^{out}(v)|$, and $|\mathcal{N}^{in}(v)|$);
  3. the relative cost of the node $c_v/|\mathcal{N}^{out}(v)|$
  4. the number of second-level neighborhood, that is, given a node v, $\sum_{u \in \mathcal{N}^{out}(v)} |\mathcal{N}^{out}(u)|$;
  5. the number of inactive nodes in the neighborhood $\mathcal{N}^{out}(v)$;
  6. the number of active nodes in the second-level neighborhood;
  7. the state of the node;
  8. an estimation of the influence power computed as the average number of nodes influenced from that given node. This value is initialized equal to the degree of the node, then it is updated each time that the node is selected. The optimistic initialization helps the exploration of the most promising nodes.
- High-order features:
  1. the size of the intersection of the neighborhood $\mathcal{N}^{out}(v)$ of the nodes considered;
  2. the size of the union of the neighborhood $\mathcal{N}^{out}(v)$ of the nodes considered;
  3. the total number of active nodes in the union of the neighborhood $\mathcal{N}^{out}(v)$;
  4. the total number of inactive nodes in the intersection of the neighborhood $\mathcal{N}^{out}(v)$.

For GNNQL, we set:

- $n = 10$;

- the GNN implementation as in [60]. Specifically, it has two dense layers of hidden nodes with the Rectified Linear Unit (ReLU) activation function [32], and the output layer uses linear activation functions;
- the graph embedding using the Diff2vec algorithm [52];
- the *Adam optimizer* to update the network weights (as described in [30]);
- the *mean square error* as a quality measure of the model (as described in [38]).

The presented parameters are tuned by following a systematic approach to strike a balance between exploration and exploitation, promote convergence, and ensure stable learning. For parameter tuning, we heavily relied on values reported in the literature as a starting point and conducted extensive experiments to find an appropriate trade-off using grid search. While there is room for further exploration and sensitivity analysis of the parameter values, we do not show results related to this topic for not increasing the length of the paper.

In TS we set π_s equal to $1/|S|$ (i.e., we use crude Monte Carlo sampling), and we apply Benders' decomposition to achieve lower computational times.

In order to better assess the performance of the proposed methods, we compare them against a set of benchmark techniques. In particular, we generalized the *billboard strategy* (BS) and the *handbill strategy* (HS) proposed in [22] for the two-stage budgeted IMP. The BS chooses the most influential nodes as seeds (i.e., the ones with the highest degrees) while HS selects the cheapest nodes (i.e., the ones with minimum ratio between cost and degree). Mimicking [22], we also define a simple decision rule that selects the best nodes according to a linear combination of AVI's first-order features. The coefficients of the combination are optimized by using particle swarm optimization. We call this solution method *combination strategy* (CS). We apply these three techniques in the adaptive setting in a greedy way: Given the available budget b_t, we select the best inactive nodes such that we spend at most $\frac{b_t}{(T-t)}$. If the resulting set of nodes to select is empty $\left(\text{i.e., the best nodes cost more than } \frac{b_t}{(T-t)}\right)$, then we select the best node whose cost is less than b_t.

Moreover, we also consider two deep RL heuristics using traditional deep neural networks, namely *proximal policy approximation* (PPO) and *advantage actor-critic* (A2C) [58].

The project has been implemented using Python3.10 and the environment has been developed using the OpenAI Gym APIs [8]. All the computational times are measured on a *Intel(R) Core(TM)* i7-5500U CPU@2.40GHz computer with 16GB of RAM and running *Ubuntu* v22.04. Gurobi v10.0.0 solves all the models via its Python3 APIs. We implement PPO, and A2C using the Python library Ray [44]. Moreover, if not otherwise stated:

- The performance of the techniques is averaged over 10 different network generations and 1000 out of sample scenarios.
- The execution times of the AVI and GNNQL models do not consider the training time for the two algorithms, these quantities will be discussed in detail when needed.

This section is organized as follows: In Subsection 5.1, we present the mathematical settings of the experiments. In Subsection 5.2, we present the results of the proposed solution methods for small instances with simple dynamics. In Subsection 5.3, we test the methods on large data instances with complex dynamics, and in Subsection 5.4, we show the results considering realistic graphs.

One important topic that is not addressed in the following subsection is convergence analysis. We report it in Appendix A since the focus of the current section is primarily to investigate the performances of the proposed techniques.

5.1 | Experimental setting

The most important characteristic of the instances is the graph G. Even though data related to some real network are available, we define a way to generate synthetic instances in order to be able to control their characteristics and to better test the proposed techniques.

We start by distinguishing between *daily active users* (DAU) and *monthly active users* (MAU). The first is the number of accounts that make access every day, while the second is computed on a monthly base. Since we are considering a marketing campaign, a time horizon of a month is reasonable, therefore we focus on the MAU statistics.

Due to the acceptable amount of data, we use the Instagram social network as a guiding example. Instagram is a photo and video-sharing social network owned by Meta Platform. Each user has the ability to share media content and it is associated with the other accounts through two sets: The one of the *followers* ($\mathcal{N}^{out}(v)$) and the one of the *following* ($\mathcal{N}^{in}(v)$). Given a user, the first set contains the users who can see his/her account. Instead, the set of accounts that the user can see constitutes the second set. In other words, for each node in the network G which represents a user, we have arcs from the node to the set of *followers* and we have arcs from the set of *following* to the node. Here the arcs represent the ability to see the contents. As of this writing, Instagram has around one billion MAU globally and around 500 million DAU [3]. The largest addressable Instagram audience size is in India, with 180 million users, followed by the United States (170 million) and Brazil (110 million). There are around

TABLE 1 Graph dimension of the greatest instance solved by the different papers.

Paper	Dataset name	V_{max}	A	Dataset name	V	A_{max}
[15]	EachMovie dataset	73k	2.8M	EachMovie dataset	73k	2.8M
[28]	High-Energy Physics	11k	53k	High-Energy Physics	11k	53k
[63]	High-Energy Physics	15k	58k	Wiki dataset	8.6k	26k
[60]	Twitter dataset	40M	1.3B	Twitter dataset	40M	1.3B
[20]	ENRON	37k	370k	ENRON	37k	370k
[27]	Twitter #datascience	25k	405k	Twitter #datascience	25k	405k
[65]	Flickr	2.5M	33M	Weibo	1.8M	308M
[40]	Pretty-Good-Privacy	11k	25k	Wiki dataset	7k	103k

Note: V_{max} represents the maximum number of nodes and A_{max} the maximum number of arcs.

TABLE 2 Parameter value used in the experiments for the CTM.

	Nodes [%]	α_k	β_k	λ_k	ζ_k
Influencer	10	5	1	0.7	0.8
Super-active	20	4	2	0.5	0.6
Active	35	3	3	0.4	0.5
Passive	35	2	4	0.1	0.1

15 million Instagram users in Italy. Since marketing campaigns are typically designed to target specific user segments based on various factors such as age, gender, occupation, income, interests, and location, attempting to include all users would be pointless. For example, restricting our interest to the female users in the age rank of 35–44 years old, there are 7.7% of all the users [56] (i.e., roughly 1 million). Moreover, considering all the MAUs as nodes is not reasonable since it can be really difficult (if not pointless) to influence users who access the social network in an almost passive way. Thus, we focus on the users who share more than 1 post per week. In particular, given the distribution of the number of posts, it seems that only 10% of accounts have such characteristics [43]. For this reason, in the computational experiments, we consider networks having nodes in the interval [10k; 100k].

Estimating the connection of each node is also challenging. We consider the Barabási-Albert random graph due to the *preferential attachment property*, that is, the more connection a node has, the more likely it is to receive new ones [1]. The Barabási-Albert graph is described by the number of nodes (V) and the number of arcs to attach from a new node to existing nodes (m). We set m to 100 and 200, considering that the average number of connections for a user is 150 [43].

The number of nodes and arcs considered are in line with the greatest instances considered in other papers dealing with the IMP. We report some of these instances in Table 1. In particular, for each paper, we report the name of the dataset with the greatest number of nodes (second column) with the relative number of nodes and arcs (third and fourth column, respectively), then we report the name of the dataset with the largest number of arcs (fifth column) together with the number of nodes and arcs (the last two columns). As the reader can notice, the experiments with the greatest number of arcs and nodes are achieved when working with social networks (Flickr and Twitter datasets). In this case, the instances reach up to 40 billion nodes and 308 million arcs, respectively.

As stated previously, the number of connections of each node affects the cost of influence. We assume that the cost c_v depends linearly on the number of connections that the node has. We calibrate it considering that one of the most paid influencers in Italy requires around 100k€ to share a single piece of content. Thus, we consider that 100k€= $\max_v c_v$ and that a user with no connection has $c_v = 0$.

Even more difficult to estimate than the previous quantities is the dynamic of the influence. In fact, this is a broad and open area of research (see, e.g., [14]). Since the proposed techniques are agnostic with respect to the dynamic of influence chosen, we test them against the models presented in Section 3, where the parameters for the CTM are reported in Table 2.

Finally, the initial budget B is a parameter that deeply affects the solution of the proposed techniques. In particular, with a high budget, the performance of all the methods tends to be similar since they can all influence the best nodes [13]. In the following, we use different quantities of budget defined as $B = \nu \max_v c_v$ where ν is a tightness factor.

5.2 | Small instances

In this section, we test the proposed techniques on small-scale graphs (up to 1000 nodes). For all experiments, we use a training set of 100 episodes for all methods, V ranging from 100 to 1000 nodes, and the initial budget equal to $\nu \max_v c_v$, with ν equal to 0.5, 1, or 2.

TABLE 3 Comparison between optimality gap and computational time of TS, AVI, and GNNQL for instances with different V, m, and S, with $T = 2$ and an ICM as diffusion process.

V	m	S	Comp time TS [s] (std.dev [s])	Gap AVI [%] (std.dev[%])	Comp time AVI [s] (std.dev [s])	Gap GNNQL [%] (std.dev[%])	Comp time GNNQL [s] (std.dev [s])
100	10	5	1.18 (0.13)	0.08 (0.01)	0.01 (0.01)	0.11 (0.01)	0.01 (0.01)
100	10	10	1.01 (0.10)	0.21 (0.03)	0.01 (0.01)	0.14 (0.02)	0.01 (0.01)
100	10	100	2.86 (0.38)	2.32 (0.12)	0.01 (0.01)	2.74 (0.03)	0.01 (0.02)
100	20	5	0.97 (0.13)	0.11 (0.04)	0.01 (0.01)	0.11 (0.01)	0.01 (0.02)
100	20	10	1.02 (0.24)	0.23 (0.19)	0.01 (0.01)	0.28 (0.02)	0.01 (0.01)
100	20	100	2.96 (0.41)	1.25 (0.5)	0.01 (0.01)	2.64 (0.11)	0.01 (0.02)
500	50	5	1.37 (0.12)	0.1 (0.3)	0.05 (0.01)	0.11 (0.01)	0.05 (0.02)
500	50	10	1.97 (0.17)	0.13 (0.02)	0.04 (0.01)	0.17 (0.01)	0.04 (0.02)
500	50	100	131.67 (12.45)	2.18 (0.09)	0.05 (0.01)	1.86 (0.06)	0.06 (0.02)
500	100	5	1.35 (0.27)	0.08 (0.01)	0.04 (0.01)	0.09 (0.01)	0.04 (0.02)
500	100	10	1.59 (0.23)	0.14 (0.05)	0.05 (0.01)	0.24 (0.01)	0.06 (0.03)
500	100	100	132.95 (10.04)	1.92 (0.20)	0.05 (0.01)	2.00 (0.08)	0.06 (0.02)
1000	100	5	1.39 (0.22)	0.13 (0.03)	0.1 (0.01)	0.09 (0.03)	0.08 (0.03)
1000	100	10	2.80 (0.32)	0.19 (0.02)	0.11 (0.01)	0.16 (0.02)	0.10 (0.05)
1000	100	100	N.A.	N.A.	0.1 (0.01)	N.A.	0.09 (0.05)
1000	200	5	1.56 (0.22)	0.07 (0.01)	0.08 (0.01)	0.14 (0.05)	0.08 (0.03)
1000	200	10	2.91 (0.35)	0.13 (0.04)	0.07 (0.01)	0.19 (0.05)	0.08 (0.02)
1000	200	100	N.A.	N.A.	0.1 (0.01)	N.A.	0.08 (0.05)

Note: The standard deviations of the values are reported within brackets.

We consider three experiments.

First, we consider a problem with $T = 2$, such that the scenarios used by TS are the only possible ones. This makes the solution of TS the optimal one, enabling us to measure the optimality gap of the AVI and GNNQL.

Second, we consider a setting in which T varies and we compare the solutions of TS (having as second stages all the possible scenarios at time T) with the ones of AVI and GNNQL. In this experimental setup, TS provides the optimal non-adaptive solution. Thus, comparing the proposed techniques against TS gives a lower bound (since AVI and GNNQL are heuristics) of the value of adaptability. The aim of this experiment is to understand if the effort in solving the adaptable problem (which usually is more complex) is worth the gain in performance.

For both experiments, we consider an ICM since it is the only diffusion model presented that satisfies the superposition principle, allowing for the usage of TS.

Third, we compare the proposed techniques against BS, HS, CS, PPO, and A2C. In this last experiment, we consider all the diffusion models presented.

The results of the first experiment are represented in Table 3. Since TS is an exact method, we report the percentage gap between its objective function and the expected reward of the other methods. We calculate using the formula

$$\frac{p^{\mathrm{TS}} - p^{Heu}}{p^{\mathrm{TS}}}, \qquad Heu \in \{\mathrm{AVI}, \mathrm{GNNQL}\}, \tag{37}$$

where p^{TS} is the optimal value of TS, and p^{Heu} is computed by averaging the results obtained by the solution of the considered heuristic over all the possible scenarios.

The trend of the computational time for the TS is clear: It increases if the number of nodes or if the number of scenarios increases. When the number of nodes is 1000 and the number of scenarios is 100, the exact solver runs out of memory, originating the non-available (N.A.) in the last rows of Table 3.

Clearly, the computational times of AVI and GNNQL are not affected by the change in the number of scenarios since both techniques do not consider them. Moreover, their computational times appear to be unaffected by the change of m but slightly increase with the dimension of the instance since more time is required to compute features and to solve the optimization model. It is interesting to notice that the average computational time of AVI is particularly small since, with the ICM, the approximation of $\hat{V}^x$ stops without considering second-order features. This shows that the iterative procedure for fitting $\hat{V}^x$ can correctly deal with easy diffusion models.

Finally, the optimality gap is minimal for both AVI and GNNQL without a clear trend. This can be due to the simple dynamic of the problem that allows both methods to easily learn the best nodes to select. More in detail, it can be noticed that AVI has a lower gap than GNNQL but it needs more time since it requires computing the optimal solution of Model (27)–(29) several times.

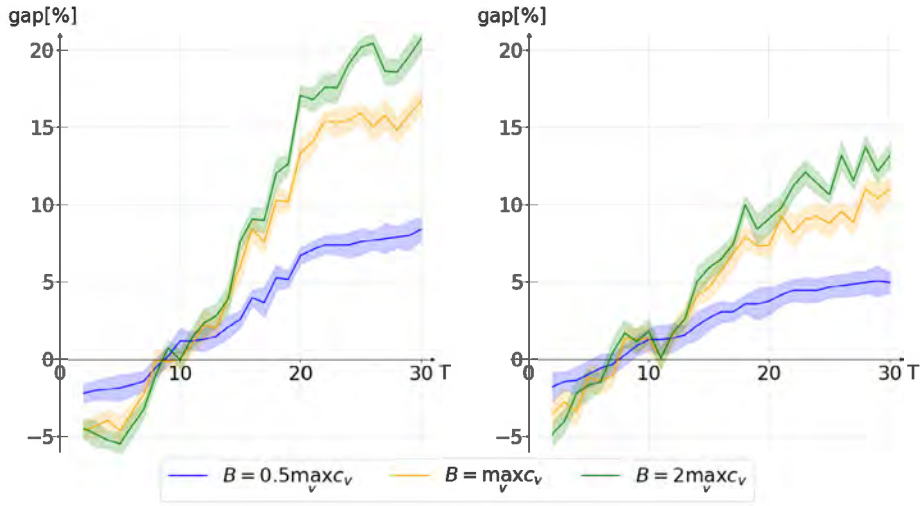

FIGURE 4 Gap between TS and AVI on the left, and TS and GNNQL on the right for different values of budget B and time horizon T.

In the second experiment, we address the setting in which AVI and GNNQL achieve the greatest gap, that is, $V = 500$, $m = 50$, $S = 100$. We report the gap computed as

$$\frac{p^{Heu} - p^{\mathrm{TS}}}{p^{Heu}}, \qquad Heu \in \{\mathrm{AVI}, \mathrm{GNNQL}\}, \tag{38}$$

for different values of T in Figure 4. As the reader can notice, up to $T = 10$ (for the AVI) and up to $T = 8$ (for the GNNQL) the gap is negative, meaning that TS achieves better results than adaptive methods. Therefore, in the considered setting with short time horizons, it is not beneficial to tackle the adaptive problem. Instead, for greater time horizons, the gap starts to be positive, and it increases up to a plateau. The value of the plateau increases as B increases. In fact, with more budget, more nodes can be selected, and more value can be extracted by using the information coming from the diffusion process.

Delving deeper into the solution, it is interesting to notice that while TS consumes all the budget in the first time period, both AVI and GNNQL invest a small part of it. This budget is mainly used to select nodes close to the ones with a high number of connections (trying to influence them indirectly). Only in the final part of the time period AVI and GNNQL directly select the most expensive nodes (if needed). This strategy and the results that they achieve confirm that the adaptive approach is better than the non-adaptive one for T sufficiently big (i.e., greater than 10 in the considered instances). Finally, we can notice that qualitatively the graphs for AVI and the one for GNNQL are similar, with the only difference being that the gaps of the GNNQL are smaller than the ones achieved by the AVI. This is related to the fact that AVI considers a different post-decision value function approximation for each time step, while for GNNQL the time step is an input parameter of the two GNNs. This provides AVI more degrees of freedom that enable it to better grasp the opportunities of the diffusion process.

In the third experiment, we compare the proposed techniques against BS, HS, CS, PPO, and A2C. We consider $T = 30$, which intuitively represents a month of a marketing campaign, and $V = \{100, 250, 500, 750, 1000\}$. For each value of V, we generate 25 instances with $m = \frac{V}{10}$, and 25 instances with $m = \frac{V}{5}$. After obtaining the results for all the methods, we calculate the percentage gap with respect to the best result achieved. Therefore, an average gap equal to zero indicates that the method is always the best or it always achieves results equal to the best one. The results are presented in Figure 5. It is worth noting that for small graphs ($V = 100$), all methods provide similar solutions; hence they have similar performance, with those based on neural networks (GNNQL, PPO, and A2C) exhibiting higher variance. However, as the graph size increases to $V = 500$, the solutions start to differ and so do the performances. Particularly, for instances with $V \geq 500$, it becomes evident that the top-performing methods are AVI and GNNQL, with AVI achieving slightly better results due to its lower variance. Both BS and HS exhibit similar performance across most instances, except for the setting with $V = 750$, where BS shows significantly poor results (its performances are, on average, 90% worst than the best method). Those poor performances of BS and HS can be attributed to the simple choices that they make. Remarkably, CS, PPO, and A2C emerge as the second-best methods, with PPO and A2C showcasing greater variance.

In terms of computational time, all methods require less than one second to compute a solution, but HS, BS, and CS stand out as the fastest due to their simple strategy. Additionally, HS and BS hold the advantage of not requiring any training. However, due to their poor performance, we do not consider them in the following subsections.

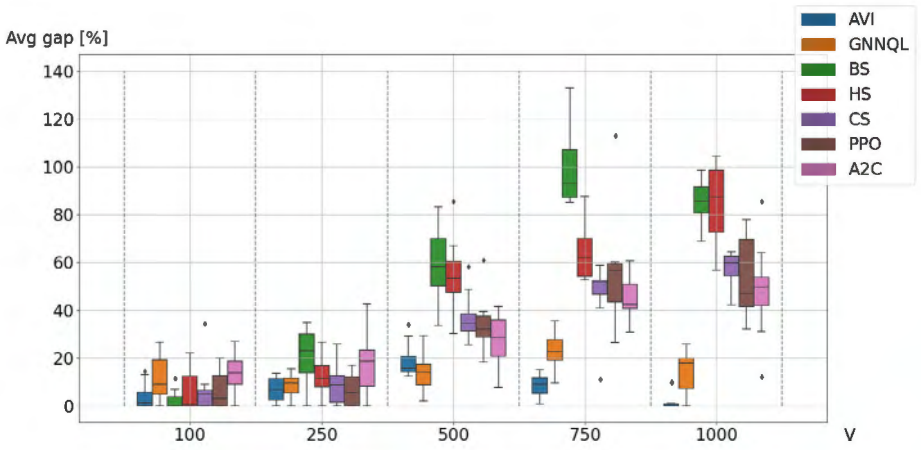

FIGURE 5 Comparison of different methods on small instances.

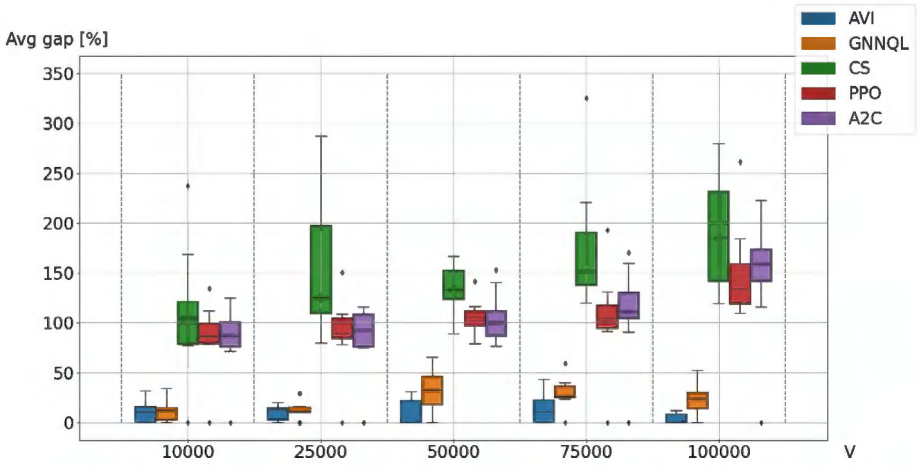

FIGURE 6 Comparison of different methods on large instances.

5.3 | Large instances

In this subsection, we investigate the performance of AVI and GNNQL on realistic-size instances. For all these experiments, we consider a training set of 1000 episodes, V ranges from $10k$ to $100k$ nodes, and the diffusion processes considered are the ones presented in Section 3. We set the initial budget $B = \nu \max_v c_v$, for ν equal to 0.8, or 1.2.

In the initial experiment, we compare the performance of AVI, GNNQL, CS, PPO, and A2C, with fixed values of $T = 30$ and a range of different values for $V = \{10{,}000, 25{,}000, 50{,}000, 75{,}000, 100{,}000\}$. For each specific value of V, we create 12 instances, resulting from three variations of both m (which can be either $\frac{V}{10}$ or $\frac{V}{5}$) and ν (which can be either 0.8 or 1.2). This totals 60 instances. As above, after obtaining the results for all the methods, we calculate the objective function gap with respect to the best one. We show the results in Figure 6. As in Subsection 5.2, AVI and GNNQL achieve the best results in all the settings (meaning that they often are the best methods among the ones considered), with AVI achieving slightly better results than GNNQL for instances greater than 50,000 nodes. Instead, CS, PPO, and A2C achieve similar results providing solutions from 90% to 150% worse than the best technique. Between them, CS shows the worst results having huge variances and never achieving the best result for instances greater than 50,000 nodes. Due to these results, we further analyze the performance of AVI and GNNQL by dividing them for the two values of ν in Tables 4 and 5, respectively.

Each table presents, for a given set of parameters V, m, and different diffusion models, the average and standard deviation of the percentage of nodes influenced at the end of the episode, along with the average and standard deviation of the computational time (in seconds) required to compute a single action.

It is worth noting that the computational time of AVI increases as V increases since the underlying optimization problem becomes bigger. Additionally, the diffusion model strongly influences the computation time, with CTM requiring more time than LTM, and LTM requiring slightly more than ICM. Also, the computational time of the GNNQL increases as V increases since more time is needed to compute the features, but GNNQL seems to be unaffected by the type of diffusion process.

TABLE 4 Percentage of nodes influenced for different graphs and diffusion process, $\nu = 0.8$.

V	m	Diffusion model	I_T/V [%] AVI	Time AVI [s]	I_T/V [%] GNNQL	Time GNNQL [s]
10,000	100	LTM	48.41 (10.23)	0.38 (0.10)	52.61 (11.12)	0.19 (0.05)
10,000	200	LTM	66.13 (12.21)	0.39 (0.14)	68.79 (8.74)	0.21 (0.16)
10,000	100	ICM	63.09 (22.08)	0.36 (0.08)	63.23 (8.48)	0.15 (0.02)
10,000	200	ICM	67.39 (20.65)	0.35 (0.24)	82.46 (6.66)	0.17 (0.12)
10,000	100	CTM	51.58 (10.41)	0.68 (0.08)	43.48 (17.84)	0.17 (0.28)
10,000	200	CTM	78.84 (12.24)	0.62 (0.44)	68.00 (7.22)	0.10 (0.09)
50,000	100	LTM	72.58 (9.80)	0.86 (0.32)	57.69 (12.31)	0.16 (0.30)
50,000	200	LTM	69.32 (9.93)	1.15 (0.33)	75.40 (8.59)	0.31 (0.44)
50,000	100	ICM	83.23 (5.64)	0.91 (0.53)	62.86 (12.46)	0.24 (0.38)
50,000	200	ICM	79.25 (7.80)	0.97 (0.45)	87.04 (7.45)	0.15 (0.24)
50,000	100	CTM	51.11 (8.24)	1.79 (1.30)	55.22 (16.72)	0.46 (0.34)
50,000	200	CTM	65.51 (6.43)	1.76 (1.11)	73.81 (9.70)	0.52 (0.28)
100,000	100	LTM	79.13 (7.23)	1.60 (0.84)	60.24 (7.92)	0.43 (0.84)
100,000	200	LTM	72.37 (12.44)	1.55 (0.95)	82.62 (9.84)	0.34 (0.32)
100,000	100	ICM	66.99 (6.17)	1.23 (0.11)	66.57 (13.32)	0.31 (0.44)
100,000	200	ICM	98.43 (2.31)	1.22 (0.58)	82.71 (8.55)	0.35 (0.51)
100,000	100	CTM	62.36 (14.32)	2.45 (0.84)	58.60 (15.99)	0.51 (0.94)
100,000	200	CTM	80.26 (8.27)	2.51 (0.69)	77.81 (6.22)	0.60 (0.89)

Note: The standard deviations are reported within brackets.

TABLE 5 Percentage of nodes influenced for different graphs and diffusion process, $\nu = 1.2$.

V	m	Diffusion model	I_T/V [%] AVI	Time AVI [s]	I_T/V [%] GNNQL	Time GNNQL [s]
10,000	100	LTM	76.74 (8.21)	0.56 (0.01)	70.33 (12.54)	0.11 (0.23)
10,000	200	LTM	86.67 (5.21)	0.42 (0.11)	84.15 (11.84)	0.10 (0.20)
10,000	100	ICM	92.51 (3.91)	0.31 (0.16)	80.11 (14.32)	0.15 (0.21)
10,000	200	ICM	82.56 (6.87)	0.26 (0.06)	81.12 (12.31)	0.15 (0.21)
10,000	100	CTM	81.49 (8.44)	0.84 (0.23)	63.59 (22.03)	0.20 (0.33)
10,000	200	CTM	77.75 (12.32)	0.77 (0.54)	80.95 (9.72)	0.14 (0.22)
50,000	100	LTM	61.81 (19.44)	0.94 (0.00)	75.85 (20.03)	0.19 (0.32)
50,000	200	LTM	98.42 (0.87)	0.80 (0.02)	86.42 (12.53)	0.15 (0.28)
50,000	100	ICM	79.28 (5.23)	0.89 (0.07)	76.87 (13.54)	0.17 (0.42)
50,000	200	ICM	86.92 (9.87)	0.69 (0.37)	89.39 (2.23)	0.13 (0.23)
50,000	100	CTM	72.32 (12.54)	2.06 (1.30)	65.43 (27.65)	0.37 (0.41)
50,000	200	CTM	69.42 (23.41)	1.98 (1.30)	88.78 (9.87)	0.58 (0.17)
100,000	100	LTM	76.91 (9.65)	1.35 (0.86)	81.59 (8.43)	0.33 (0.16)
100,000	200	LTM	74.27 (9.98)	1.33 (0.49)	88.04 (6.49)	0.34 (0.38)
100,000	100	ICM	90.58 (2.71)	1.05 (0.98)	89.81 (5.33)	0.36 (0.72)
100,000	200	ICM	85.59 (7.82)	1.08 (0.82)	68.46 (12.15)	0.28 (0.54)
100,000	100	CTM	67.58 (21.09)	2.30 (1.66)	77.16 (12.82)	0.56 (0.61)
100,000	200	CTM	98.68 (0.56)	2.76 (1.58)	86.73 (5.32)	0.60 (0.34)

Note: The standard deviations are reported within brackets.

Regarding the objective function, we can observe that the results with more budget ($\nu = 1.2$) are better than those achieved with less budget ($\nu = 0.8$). Moreover, the results with $m = 200$ are better than the ones with $m = 100$ since the greater the quantity of edges in the network, the easier it is to influence more nodes.

Delving deeper into the solutions, it is interesting to note that if $m = 200$, the solutions of both methods tend to consider just the nodes with more connections; for this reason, with $m = 200$, the gap between the AVI and GNNQL is smaller than with $m = 100$. Moreover, the main difference between the two methods can be seen when $m = 100$ and $\nu = 0.8$. This is the most challenging condition in which we test the proposed methods. In that setting, the performances of the AVI are far better than the ones achieved by GNNQL. It is interesting to note that AVI achieves these results by value function approximations that rarely are of the third order.

In summary, the proposed heuristics show good performances in the considered instances. Moreover, they prove to be robust with respect to different diffusion processes. Care must be taken to ensure that these results can be replicated in practice due to

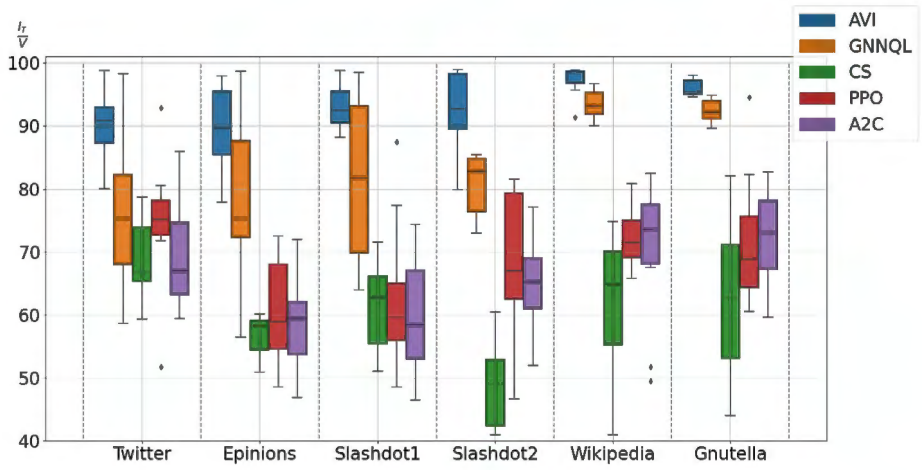

FIGURE 7 Comparison of different methods on real instances.

several characteristics that the environment does not address. Nevertheless, it seems clear that, given a realistic environment, the proposed techniques provide a reasonable way to compute the set of nodes to be selected in a reasonable amount of time.

5.4 | Realistic instances

In this subsection, we evaluate the performance of both AVI and GNNQL using realistic instances. For all these experiments, we consider a training set of 800 episodes.

We consider six real datasets[2]:

- The Twitter circles graph dataset: Taken from the popular social network, this dataset includes lists of connected users, it has 81,306 nodes and 1,768,149 arcs [42].
- The Epinions social network graph dataset: It was a who-trust-whom online social network that collected general consumer reviews sampled from online e-commerce. It has 75,879 nodes and 508,837 arcs [51].
- The Slashdot network 1: It is a sample of a social news website obtained in November 2008. The data are related to Slashdot Zoo an online feature that allows users to tag each other as friends or foes. It contains 77,360 nodes and 905,468 arcs [36].
- The Slashdot network 2: It is a snapshot of the Slashdot Zoo social network taken in February 2009. The dataset comprises 82,168 nodes and 948,464 arcs [36].
- The Wikipedia vote network: Based on the popular free encyclopedia elections to promote users to administrators. The dataset represents the users participating in the elections of January 2008 and the relative votes. It consists of 7115 nodes and 103,689 arcs [34].
- The Gnutella graph: It is a snapshot of the Gnutella peer-to-peer file-sharing network from August 2002. In this dataset, nodes represent peers, and arcs represent file-sharing connections. It has 10,876 nodes and 39,994 arcs [35].

Using these networks, we consider the diffusion processes presented in Section 3. Similar to Subsection 5.3, we consider the tightness factor ν equal to 0.8, or 1.2. The comparison between AVI, GNNQL, CS, PPO, and A2C in terms of I_T/V is shown in Figure 7.

Also in these instances, both AVI and GNNQL outperform CS, PPO, and A2C. Specifically, AVI achieves the best results with the highest average I_T/V, and it manages to influence almost all the nodes in several scenarios. Both PPO and A2C demonstrate similar performances and constitute the second-best methods. Instead, CS performs the worst among the algorithms, often having the lowest average and exhibiting significant variance in several instances.

In general, all the methods exhibit improved performance in instances with a higher ratio of arcs to nodes, as this facilitates diffusion and simplifies the problem. It is interesting to note that the results on realistic graphs are superior to those obtained with the large instance presented in Subsection 5.3. This difference is likely due to the degree distribution, which makes the realistic graph more favorable.

To better analyze the performance of AVI and GNNQL, we report the average I_T/V, the time to compute the action (in seconds), and the training time (in hours) in Table 6. The standard deviation of these values is reported within brackets.

[2]Downloaded from https://snap.stanford.edu/data/.

TABLE 6 Percentage of nodes influenced by AVI and GNNQL on real instances.

		AVI			GNNQL		
ν	Diffusion	I_T/V[%]	Action comp. [s]	Training [h]	I_T/V[%]	Action comp. [s]	Training [h]
0.8	LTM	88.55 (5.94)	0.94 (2.26)	7.27 (0.63)	78.80 (12.76)	0.22 (0.19)	1.68 (1.06)
1.2	LTM	95.08 (4.59)	0.81 (1.57)	6.16 (0.56)	92.21 (13.14)	0.27 (0.12)	2.08 (0.40)
0.8	ICM	94.14 (4.44)	0.71 (1.08)	5.44 (3.36)	84.67 (8.07)	0.24 (0.01)	1.90 (0.33)
1.2	ICM	97.87 (4.54)	0.77 (2.02)	6.00 (1.32)	88.02 (7.06)	0.24 (0.04)	1.81 (0.20)
0.8	CTM	74.57 (5.75)	1.49 (3.97)	11.55 (3.41)	71.25 (13.97)	0.53 (0.21)	4.05 (0.12)
1.2	CTM	83.89 (5.92)	1.64 (2.23)	12.75 (2.37)	76.28 (13.60)	0.57 (0.23)	4.34 (0.75)

Note: The standard deviations are reported within brackets.

As the reader can notice, the ratio I_T/V increases as ν increases, since the greater the budget, the easier is to influence more nodes. Nevertheless, the increment in I_T/V is small for instances considering the ICM and relatively large for those considering the CTM. This means that the marginal economic value of the budget is influenced by the type of diffusion process considered, and the same holds for the strategic decision about the quantity of budget to allocate in the marketing campaign. Moreover, in absolute terms, the greatest value of I_T/V is achieved for the instances considering the ICM, and the lowest ones in the instances considering CTM. This effect is due to the difference in the complexity of the dynamics.

The time to compute the action (columns *single step*) is below 2 s for both methods. This means that both AVI and GNNQL can be applied in the real field and that they can also be used for building decision support systems in which the decision-maker may interact with the solver to perform what-if analysis.

Instead, the training times (reported columns *training*) are not negligible: While the maximum training time is 4 h for GNNQL, AVI may require up to 12 h. It is interesting to notice that the diffusion model strongly influences both the time to compute the action and the training time. This is an important analysis to perform since in the real field the diffusion model is not known. Moreover, this analysis paves the way for future investigation on how the methods developed for the IMP work when different diffusion models are considered.

6 | CONCLUSIONS

The paper presents and compares a set of algorithms for the *adaptive budgeted influence maximization problem*. In particular, we propose a customized approximated value iteration leveraging mixed integer programming and a customized approximate policy iteration exploiting new advances in graph neural networks.

The techniques achieve results close to the optimal ones (computed with a stochastic two-stage model) in an ad-hoc setting. Moreover, through a wide set of computational experiments, we prove that they achieve good results on realistic graphs, being able to effectively adapt to different topologies and diffusion processes. Since we have used synthetic data, it would not be very "scientific" to conclude that the proposed methods work well also in a real environment. Nevertheless, they can be considered at least from a computational point of view.

In conclusion, it is worth noting that despite the use case considering social networks in their web meaning, the proposed algorithms can be applied to other different contexts (e.g., awareness spreading, [10]).

Future works will address techniques to combine information from the value function in two-stage problems in order to reduce their inherent myopic behavior, consider continuous node states (to model more shades of belief), address more advanced diffusion processes (incorporating changes in the number of nodes and arcs, addressing the role of groups [71], considering negative influences [5, 12], etc.), and study the robustness of these methods when the training and testing are performed using different diffusion models.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Edoardo Fadda https://orcid.org/0000-0002-5599-6349
Evelina Di Corso https://orcid.org/0000-0002-3988-3512

REFERENCES

[1] R. Albert and A.-L. Barabási, *Statistical mechanics of complex networks*, Rev. Mod. Phys. **74** (2002), 47–97.

[2] P. Almasan, J. Suárez-Varela, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio, *Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case*, Comput. Commun. **196** (2022), 184–194.
[3] Backlinko.com. Instagram demographic statistics. 2022.
[4] A. Baldo, M. Boffa, L. Cascioli, E. Fadda, C. Lanza, and A. Ravera, *The polynomial robust knapsack problem*, Eur. J. Oper. Res. **305** (2023), 1424–1434.
[5] S. Bharathi, D. Kempe, and M. Salek, "*Competitive influence maximization in social networks*," *Lecture notes in computer science*, Springer, Berlin Heidelberg, 2007, pp. 306–311.
[6] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. Proceedings of the twenty-fifth annual ACM-SIAM symposium on discrete algorithms. Society for Industrial and Applied Mathematics, p. 946–957. 2013.
[7] P. Brandimarte, *From shortest paths to reinforcement learning*, Springer International Publishing, Berlin, Heidelberg, 2021.
[8] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. 2016.
[9] business.trustedshops. Gruppi utenti e social media. 2021.
[10] H. Chen, W. Qiu, H.-C. Ou, B. An, and M. Tambe. Contingency-aware influence maximization: A reinforcement learning approach. 2021.
[11] T. Chen, J. Guo, and W. Wu, *Adaptive multi-feature budgeted profit maximization in social networks*, Soc. Netw. Anal. Min. **12** (2022), 164.
[12] W. Chen, A. Collins, R. Cummings, T. Ke, Z. Liu, D. Rincon, X. Sun, Y. Wang, W. Wei, and Y. Yuan. Influence maximization in social networks when negative opinions may emerge and propagate. Proceedings of the 2011 SIAM international conference on data mining. Society for Industrial and Applied Mathematics, pp. 379–390. 2011.
[13] J.-F. Cordeau, F. Furini, and I. Ljubić, *Benders decomposition for very large scale partial set covering and maximal covering location problems*, Eur. J. Oper. Res. **275** (2019), 882–896.
[14] A. Cuzzocrea, C. K. Leung, D. Deng, J. J. Mai, F. Jiang, and E. Fadda, *A combined deep-learning and transfer-learning approach for supporting social influence prediction*, Procedia Comput Sci **177** (2020), 170–177.
[15] P. Domingos and M. Richardson. Mining the network value of customers. Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining. Association for Computing Machinery (ACM), pp. 57–66. 2001.
[16] C. Gao, S. Gu, J. Yu, H. Du, and W. Wu, *Adaptive seeding for profit maximization in social networks*, J. Glob. Optim. **82** (2021), 413–432.
[17] D. Golovin and A. Krause, *Adaptive submodularity: Theory and applications in active learning and stochastic optimization*, J. Artif. Intell. Res. **42** (2011), 427–486.
[18] P. Goyal and E. Ferrara, *Graph embedding techniques, applications, and performance: A survey*, Knowl.-Based Syst. **151** (2018), 78–94.
[19] E. Güney, *On the optimal solution of budgeted influence maximization problem in social networks*, Oper. Res. **19** (2017), 817–831.
[20] E. Güney, M. Leitner, M. Ruthmair, and M. Sinnl, *Large-scale influence maximization via maximal covering location*, Eur. J. Oper. Res. **289** (2021), 144–164.
[21] K. Han, K. Huang, X. Xiao, J. Tang, A. Sun, and X. Tang, *Efficient algorithms for adaptive influence maximization*, Proc VLDB Endow **11** (2018), 1029–1040.
[22] S. Han, F. Zhuang, Q. He, and Z. Shi, "*Balanced seed selection for budgeted influence maximization in social networks*," *Advances in knowledge discovery and data mining*, Springer International Publishing, Berlin, Heidelberg, 2014.
[23] H. V. Hasselt, A. Guez, and D. Silver, *Deep reinforcement learning with double q-learning*, Proc AAAI Conf Artif Intell **30** (2016), 2094–2100.
[24] K. Huang, J. Tang, K. Han, X. Xiao, W. Chen, A. Sun, X. Tang, and A. Lim, *Efficient approximation algorithms for adaptive influence maximization*, Vldb J. **29** (2020), 1385–1406.
[25] K. Huang, Y. Wu, X. Zhang, S. Tu, Q. Wu, M. Wang, and H. Wang. Provably efficient reinforcement learning for online adaptive influence maximization. 2022.
[26] C. O. Huelsnitz, R. E. Jones, J. A. Simpson, K. Joyal-Desmarais, E. C. Standen, L. A. Auster-Gussman, and A. J. Rothman, *The dyadic health influence model*, Personal. Soc. Psychol. Rev. **26** (2021), 3–34.
[27] M. Kahr, M. Leitner, M. Ruthmair, and M. Sinnl, *Benders decomposition for competitive influence maximization in (social) networks*, Omega **100** (2021), 102264.
[28] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining. Association for Computing Machinery (ACM), p. 137–146. 2003.
[29] D. Kempe, J. Kleinberg, and É. Tardos, "*Influential nodes in a diffusion model for social networks*," *Automata, languages and programming*, Springer, Berlin Heidelberg, 2005, pp. 1127–1138.
[30] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. 2014.
[31] S. Kumar, A. Mallik, and B. Panda, *Influence maximization in social networks using transfer learning via graph-based LSTM*, Expert Syst. Appl. **212** (2023), 118770.
[32] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature **521** (2015), 436–444.
[33] K. M. Lee, S. G. Subramanian, and M. Crowley. Investigation of independent reinforcement learning algorithms in multi-agent environments. 2021.
[34] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. Proceedings of the SIGCHI conference on human factors in computing systems. Association for Computing Machinery (ACM), p. 1361–1370. 2010.
[35] J. Leskovec, J. Kleinberg, and C. Faloutsos, *Graph evolution: Densification and shrinking diameters*, ACM Trans. Knowl. Discov. Data **1** (2007), 2–43.
[36] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, *Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters*, Internet Math. **6** (2009), 29–123.
[37] H. Li, M. Xu, S. S. Bhowmick, C. Sun, Z. Jiang, and J. Cui. Disco: Influence maximization meets network embedding and deep learning. 2019.
[38] K. Liano, *Robust error measure for supervised neural network learning with outliers*, IEEE Trans. Neural Netw. **7** (1996), 246–250.
[39] J. Luo and R. Yu, *Follow the heart or the head? The interactive influence model of emotion and cognition*, Front. Psychol. **6** (2015), 573.
[40] L. Ma, Z. Shao, X. Li, Q. Lin, J. Li, V. C. M. Leung, and A. K. Nandi, *Influence maximization in complex networks by using evolutionary deep reinforcement learning*, IEEE Trans Emerg Topics Comput Intell **7** (2023), 995–1009.
[41] S. Manchanda, A. Mitaal, A. Dhawan, S. Medya, S. Ranu, and A. Singh, "*Gcomb: Learning budget-constrained combinatorial algorithms over billion-sized graphs*," *Advances in neural information processing systems*, Vol **33**, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (eds.), Curran Associates, Inc, Norwalk, 2020, pp. 20000–20011.
[42] J. McAuley and J. Leskovec. Learning to discover social circles in ego networks. Proceedings of the 25th international conference on neural information processing systems-volume 1. NIPS'12. Red Hook, NY, USA: Curran Associates Inc., p. 539–547. 2012.
[43] Mention.com. Instagram followers. 2022.

[44] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan, and I. Stoica. Ray: A distributed framework for emerging ai applications. 2017.
[45] E. Mossel and S. Roch. On the submodularity of influence in social networks. Proceedings of the thirty-ninth annual ACM symposium on theory of computing. Association for Computing Machinery (ACM), p. 128–134. 2007.
[46] nextredigital. Esempi di social media marketing. 2021.
[47] H. Nguyen and R. Zheng, *On budgeted influence maximization in social networks*, IEEE J Sel Areas Commun **31** (2013), 1084–1094.
[48] W. B. Powell, *Approximate dynamic programming*, John Wiley & Sons, Inc, Hoboken, New Jersey, 2011.
[49] S. Raghavan and R. Zhang, *A branch-and-cut approach for the weighted target set selection problem on social networks*, INFORMS J Optim **1** (2019), 304–322.
[50] S. Raghavan and R. Zhang, *Weighted target set selection on trees and cycles*, Networks **77** (2020), 587–609.
[51] M. Richardson, R. Agrawal, and P. Domingos, "*Trust management for the semantic web*," *Lecture notes in computer science*, Springer, Berlin Heidelberg, 2003, pp. 351–368.
[52] B. Rozemberczki and R. Sarkar, "*Fast sequence-based embedding with diffusion graphs*," *Complex networks IX*, Springer International Publishing, Berlin, Heidelberg, 2018, pp. 99–107.
[53] A. Saito. Curriculum learning based on reward sparseness for deep reinforcement learning of task completion dialogue management. Proceedings of the 2018 EMNLP workshop SCAI: The 2nd international workshop on search-oriented conversational AI. Association for Computational Linguistics. 2018.
[54] D. Schuler, *Social computing*, Commun. ACM **37** (1994), 28–29.
[55] L. Seeman and Y. Singer. Adaptive seeding in social networks. Paper presented at: 2013 IEEE 54th annual symposium on foundations of computer science. Institute of Electrical and Electronics Engineers (IEEE), pp. 459–468. 2013.
[56] Statista.com. Distribution of instagram users. 2022.
[57] R. Stekolshchik. Some approaches used to overcome overestimation in deep reinforcement learning algorithms. 2020.
[58] R. Sutton and A. Barto, *Reinforcement learning: An introduction*, IEEE Trans. Neural Netw. **9** (1998), 1054.
[59] Y. Tang, X. Xiao, and Y. Shi. Influence maximization. Proceedings of the 2014 ACM SIGMOD international conference on Management of Data. Association for Computing Machinery (ACM). 2014.
[60] S. Tian, S. Mo, L. Wang, and Z. Peng, *Deep reinforcement learning-based approach to tackle topic-aware influence maximization*, Data Sci Eng **5** (2020), 1–11.
[61] G. Tong and R. Wang, *On adaptive influence maximization under general feedback models*, IEEE Trans. Emerg. Top. Comput. **10** (2022), 463–475.
[62] G. Tong, R. Wang, Z. Dong, and X. Li, *Time-constrained adaptive influence maximization*, IEEE Trans Comput Soc Syst **8** (2021), 33–44.
[63] G. Tong, W. Wu, S. Tang, and D.-Z. Du, *Adaptive influence maximization in dynamic social networks*, IEEE/ACM Trans. Netw **25** (2017), 112–125.
[64] S. Vaswani and L. V. S. Lakshmanan. Adaptive influence maximization in social networks: Why commit when you can adapt? 2016.
[65] C. Wang, Y. Liu, X. Gao, and G. Chen, *A reinforcement learning model for influence maximization in social networks*, Database Syst Adv Appl (2021), 701–709.
[66] S. Wang, Z. Xu, and V.-A. Truong. Beyond adaptive submodularity: Adaptive influence maximization with intermediary constraints. 2019.
[67] T. Wang, J. He, and X. Wang, *An information spreading model based on online social networks*, Phys A: Stat Mech Appl **490** (2018), 488–496.
[68] H.-H. Wu and S. Küçükyavuz, *A two-stage stochastic programming approach for influence maximization in social networks*, Comput. Optim. Appl. **69** (2017), 563–595.
[69] W. Xu and W. Wu, *Optimal social influence*, Springer International Publishing, Berlin, Heidelberg, 2020.
[70] J. Yuan and S.-J. Tang. Adaptive discount allocation in social networks. Proceedings of the 18th ACM international symposium on Mobile ad hoc networking and computing. Association for Computing Machinery (ACM), pp. 1–10. 2017.
[71] J. Zhu, S. Ghosh, and W. Wu, *Group influence maximization problem in social networks*, IEEE Trans Comput Soc Syst **6** (2019), 1156–1164.

How to cite this article: E. Fadda, E. D. Corso, D. Brusco, V. S. Aelenei, and A. B. Rares, *Math-based reinforcement learning for the adaptive budgeted influence maximization problem*, Networks.. (2023), 1–23. https://doi.org/10.1002/net.22206

APPENDIX A: CONVERGENCE

In this section, we show the convergence analysis of AVI, GNNQL, PPO, and A2C. Convergence can be addressed in several ways (e.g., considering the change of the parameters of the value function estimation, the gradient variation in the optimization of the GNN parameters, etc.). Due to the heterogeneity of the solution techniques considered, we check convergence by the ratio

$$\frac{of_{e+\tau} - of_e}{of_e}, \tag{A1}$$

where of_e is the estimation of the expected objective function computed over 100 scenario paths using an agent trained with e episodes. Given τ, Equation (A1) is the marginal relative gain of adding τ more episodes in the training set. We represent the average values and the 90% confidence intervals (computed on a set of 10 representative instances) for different values of e, for both the large and real instances in Figure A1. As expected, in both types of instances, the values decrease and tend to zero, meaning that the utility of adding more episodes decreases and converges to zero. It can be noticed that GNNQL converges

faster than AVI, which converges faster than both PPO and A2C. Nevertheless, all the methods seem to converge for a number of episodes less than 1000 for the large instances and less than 600 for the real ones.

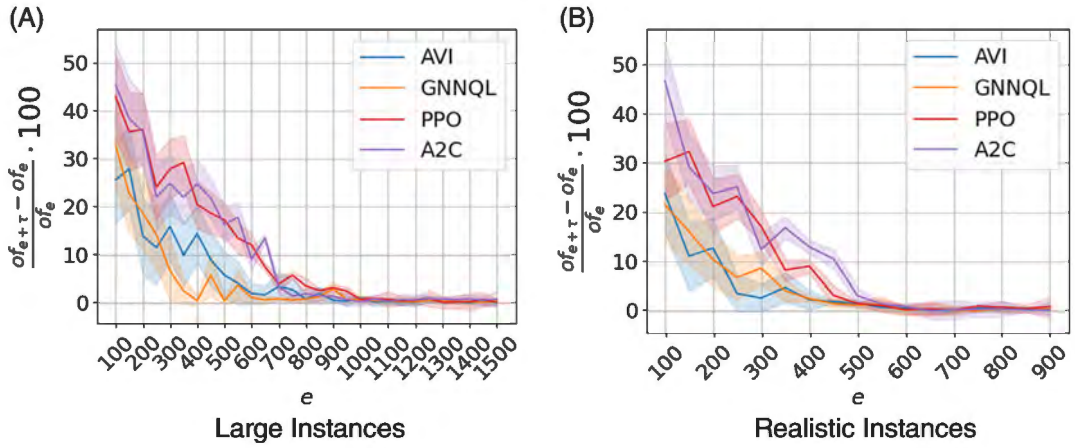

FIGURE A1 Convergence analysis for τ =50 episodes.