Influence Maximization in Complex Networks by Using Evolutionary Deep Reinforcement Learning

Lijia Ma, Zengyang Shao, Xiaocong Li, Qiuzhen Lin, *Member, IEEE*, Jianqiang Li, *Member, IEEE*, Victor C. M. Leung, *Life Fellow, IEEE*, and Asoke K. Nandi, *Fellow, IEEE*

***Abstract*—Influence maximization (IM) in complex networks tries to activate a small subset of seed nodes that could maximize the propagation of influence. The studies on IM have attracted much attention due to their wide applications such as item recommendation, viral marketing, information propagation and disease immunization. Existing works mainly model the IM problem as a discrete optimization problem, and use either approximate or meta-heuristic algorithms to address this problem. However, these works are hard to find a good tradeoff between effectiveness and efficiency due to the NP-hard and large-scale network properties of the IM problem. In this article, we propose an evolutionary deep reinforcement learning algorithm (called EDRL-IM) for IM in complex networks. First, EDRL-IM models the IM problem as a continuous weight parameter optimization of deep Q network (DQN). Then, it combines an evolutionary algorithm (EA) and a deep reinforcement learning algorithm (DRL) to evolve the DQN. The EA simultaneously evolves a population of individuals, and each of which represents a possible DQN and returns a solution to the IM problem through a dynamic markov node selection strategy, while the DRL integrates all information and network-specific knowledge of DQNs to accelerate their evolution. Systematic experiments on both benchmark and real-world networks show the superiority of EDRL-IM over the state-of-the-art IM methods in finding seed nodes.**

***Index Terms*—Complex networks, influence maximization, deep reinforcement learning, evolutionary algorithm, optimization.**

I. Introduction

THE last decades have witnessed that the advance of Internet and Web2.0 greatly promotes information communication and propagation, and produces tremendous applications for various complex systems, especially for online social systems such as Facebook, Wechat, Tencent, Tmail, JDmail, Twitter and Microblog. For example, in Internet, individuals can easily communicate with each other by various online social systems, and these systems can be used as online platforms for the promotion and recommendation of news, brands, ideas, products and behaviors via the "word-of-mouth" and "viral marketing" effects [1], [2]. Complex networks are very simple but effective models to represent these systems, where nodes denote entities while links represent information communications between entities, and the network analyses are essential to understanding the information propagation and application extension of systems [1], [3], [4].

Influence maximization (IM) in complex networks is a fundamental problem for the understanding of information propagation in real-world systems, which tries to activate a small subset of seed nodes that could maximize the propagation of information or influence [5]–[7]. This problem was first proposed by Domingos and Richardson to analyze the influence of market users in marketing fields [8], and then it was introduced into the field of complex networks by Kempe *et al.* [6]. Then, the studies on IM in networks have attracted much attention due to their wide applications in various domains such as item recommendation, viral marketing, rumor blocking, political election, traffic dispatch, information diffusion, truth discovery and disease immunization. For example, in the recent COVID-19 pandemic, the study on the IM can effectively identify super spreaders, and achieve targeted immunization to reduce the spread of COVID-19 [9]–[11]. However, the solution to the IM problem is challenging as this problem is a discrete NP-hard problem under various diffusion models such as the independent cascade (IC) [12], linear threshold (LT) [13] and time-aware diffusion (TD) [14] models, and the evaluation of seed influence is computationally complex. Moreover, the effectiveness and scalability of the solution are hard to be maintained with the increased scale, heterogeneity and accuracy of real-world systems [6].

To address the aforementioned issues, many works have been proposed for solving the IM problem in complex networks. Most of them first model the IM problem as a discrete seed optimization problem under a specific influence diffusion model, and then present either approximate or meta-heuristic algorithms to address this optimization problem. The approximate algorithms attempt to find quickly near-optimal solutions with an approximation guarantee toward the global optimum by using individual-based greedy strategies. Representative approximate algorithms include Greedy [6], CELF [6], CELF++ [15], CGA [16] and

Manuscript received 29 July 2021; revised 24 October 2021; accepted 3 November 2021. Date of publication 13 January 2022; date of current version 24 July 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 62173236, 61803269, 61876110, 61806130, 61976142, U1713212, 62072315 and 61836005, in part by the Natural Science Foundation of Guangdong Province under Grant 2020A1515010790, and in part by the Technology Research Project of Shenzhen City under Grant JCYJ20190808174801673. *(Corresponding author: Qiuzhen Lin.)*

Lijia Ma, Zengyang Shao, Xiaocong Li, Qiuzhen Lin, Jianqiang Li, and Victor C.M. Leung are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: omegamalj@gmail.com; 1910272050@email.szu.edu.cn; 2016150067@email.szu.edu.cn; qiuzhlin@szu.edu.cn; lijq@szu.edu.cn; vleung@ece.ubc.ca).

Asoke K. Nandi is with the Department of Electronic and Electrical Engineering, Brunel University London, UB8 3PH Uxbridge, U.K., and with the Shenzhen University, Shenzhen 518060, China (e-mail: Asoke.Nandi@brunel.ac.uk).

Digital Object Identifier 10.1109/TETCI.2021.3136643

CI [5]. Greedy [6] first calculated the marginal gain of each node under the IC model, and then iteratively selected the node with the maximum marginal gain into the seed set. It can guarantee the solution performance approximated to at least 63% of optimal for the IM problem. CELF [6] combined the Greedy algorithm with submodularity and lazy-forward optimization to improve the solution efficiency. CELF++ [15] improved the performance of CELF by introducing the submodularity of the spread function into IC model to decrease the repetitive influence calculation. CGA [16] and CI [5] introduced the community information and network percolation into the IM problem, and presented community-aware and collective influence-aware greedy algorithms to select seed nodes. These works [5], [6], [15], [16] were further extended by taking the GPU acceleration [17], signed influence [18], adaptive influence [19], target-aware holistic influence [20] and information privacy [21] into consideration. Although they facilitate solution efficiency, these approximate algorithms [5], [6], [15], [16] are easy to get trapped into local optima.

Compared with the approximate algorithms, the metaheuristic algorithms try to find the global optimum by using population-based evolutionary strategies inspired by natural phenomena. Representative metaheuristic algorithms for the IM problem include CMA-IM [7], DPSO-IM [22], GWO-IM [23], DDSE [24], MA-IM$_{\text{multi}}$ [25] and IICEA [26]. CMA-IM [7] first modeled the IM problem as an optimization of 2-hop influence spread, and then presented a memetic algorithm with a genetic algorithm and a local search to find seed nodes. Following CMA-IM, DPSO-IM [22] and GWO-IM [23] employed the discrete particle swarm and grey wolf optimizers to solve the IM problem, respectively. DDSE [24] combined a degree-aware greedy algorithm and an evolutionary algorithm to quickly find influential nodes. MA-IM$_{\text{multi}}$ [25] extended CMA-IM for tackling the IM problem by taking the multiplex links of networks into consideration. IICEA [26] combined a local-global influence indicator and an evolutionary algorithm for solving the IM problem, which enables to promote the evolution of solutions. These meta-heuristic algorithms [7], [22]–[26] facilitate solution exploration, but they lack efficiency and are hard to find good solutions within limited computational capacity for large-scale networks.

Deep reinforcement learning (DRL) has recently attracted much attention in the tasks of network analyses such as the minimum vertex cover, maximum cut, traveling salesman and IM problems due to its good performance in solving combinatorial optimization problems [27]–[31]. DRL incorporates a deep learning framework and an agent markov decision process into the solution of problems, which enables the agent to learn automatically the strategy that optimizes the objective [28]–[31]. Representative DRL works for network analyses include S2V-DQN [28], GCOMB [29], DISCO [30] and FINDER [31]. S2V-DQN [28] first learned the low-dimensional node features of networks through structure2vec (S2V), and then trained a deep Q network (DQN) through a DRL with these node features to solve the minimum vertex cover, maximum cut and IM problems in complex networks. GCOMB [29] extended S2V-DQN by using a graph convolutional network to prune bad nodes and learning the low-dimensional node features in a supervised manner, which enables to reduce the computational complexity for training the DQN. DISCO [30] further extended S2V-DQN by using sub-graphs to train the DQN, which enables to solve the combinational optimization problems for large-scale networks. FINDER [31] incorporated inductive network presentation learning to represent network states, and employed a DRL to solve the IM problem in networks. Although some progress has been made in using the DRLs to tackle the IM problem, these works find it hard to obtain a good tradeoff between effectiveness and efficiency due to the limited training and optimization of the DRLs.

In this article, we propose a metaheuristic evolutionary DRL algorithm (called EDRL-IM) for solving the IM problem in complex networks, aiming to train efficiently an optimal DQN that is used to make decisions for the selection of seed nodes. To this end, EDRL-IM first combines a DQN and a dynamic markov node selection strategy to select seed nodes, and then models the IM problem as a continuous weight parameter optimization of DQN. Then, EDRL-IM combines an evolutionary algorithm (EA) and a DRL algorithm to optimize the DQN. More specifically, EDRL-IM designs the EA to evolve a population of individuals simultaneously, each of which represents a DQN that can be used to make decisions for the selection of seed nodes, while it devises the DRL to accelerate the evolution by integrating all network-specific knowledge and DQNs' training information. EA and DRL facilitate exploration and exploitation for continuous optimization problems, respectively, which enable EDRL-IM to obtain a good tradeoff between effectiveness and efficiency. The main contributions of this article are shown as follows:

1) We model the IM problem as a continuous parameter optimization of DQN, and combine a DQN with a dynamic node activating strategy to select seed nodes. These guarantee the effectiveness of the proposed metaheuristic algorithm in solving the IM problem.
2) We propose the novel metaheuristic EDRL-IM algorithm for solving the continuous parameter optimization of DQN. EDRL-IM combines an EA algorithm and a DRL algorithm, and incorporates the network-specific knowledge and DQNs' training information. These enable EDRL-IM to obtain a good tradeoff between effectiveness and efficiency in optimizing the DQN.
3) Extensive experiments on the two GN and two LFR benchmark networks and ten real-world networks show that EDRL-IM outperforms the state-of-the-art IM algorithms in terms of effectiveness and efficiency.

The rest of the article is organized as follows. Section II gives the formulation of the IM problem. Section III and IV introduce our continuous DQN parameter optimization model and algorithm solution (EDRL-IM), respectively. Section V shows experimental results, while Section VI provides some concluding remarks and future works.

II. Problem Formulation

In this section, the influence propagation model is first introduced, and then the IM problem is presented and formulated.

Notations: We use italic lower-case, italic upper-case and block upper-case letters to represent scalars, vectors and

matrices, respectively, while adopt decorated letters to denote sets. Moreover, we use $\cdot$ in the index to represent the index set that spans all the row or column indices of a matrix. In addition, we use the operator $|.|$ to evaluate the number of elements in a set.

We let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be an undirected complex network, which consists of sets of nodes $\mathcal{V} = \{v_i\}_n$ and edges $\mathcal{E} = \{e_{ij}\}_m$, where e_{ij} denote an edge between nodes v_i and v_j, while n and m are the numbers of nodes and edges of $\mathcal{G}$, respectively. The edges $\mathcal{E}$ of $\mathcal{G}$ can be mathematically represented by an adjacent matrix $\mathrm{B} = [b_{ij}]_{i,j\in\mathcal{V}}$, where each element $b_{ij} \in \{0, 1\}$ denotes a possible link state between nodes v_i and v_j. Formally, b_{ij} is represented as follows:

$$b_{ij} = \begin{cases} 1 & \text{if nodes } v_i \text{ and } v_j \text{are linked,} \\ 0 & \text{otherwise.} \end{cases}$$

For each node $i \in \mathcal{V}$, we let $\mathcal{T}_i = \{j \in \mathcal{V} | b_{ij} = 1\}$ denote its neighbor that is composed of the set of nodes linked with i.

A. Influence Propagation Model

The influence propagation model describes the influence propagation (node activating) processes through a complex network triggered by an initial node set $\mathcal{S} \in \mathcal{V}$. Classical influence propagation models include the IC [1], [25], LT [13] and TD [6], [14]. Here, we mainly consider the IC model, and our work can be extended to the LT and TD models.

The IC model is motivated by the "word-of-mouth" effects, i.e., individuals may be influenced by the opinions, ideas and decisions of their friends, and influenced individuals may further influence their neighbors. The main ideas behind the IC model are that each node, once activated, has the ability to active its neighbors based on the propagation probability of edges and this propagation is iteratively and independently occurred until no further node is activated. Accordingly, in the IC model, each node $v_i \in \mathcal{V}$ has two possible states $s_i \in \{1, 0\}$, where $s_i = 1$ represents an active node i, and $s_i = 0$ denotes otherwise. Moreover, each $e_{ij} \in \mathcal{E}$ has a probability p_{ij} which determines the propagation probability of node v_i to active node v_j.

Given an initial seed set $\mathcal{S}$, the IC model works as follows:

Initialization: We let $\mathcal{S}_t$ be the active node set after the tth propagation step, and let $\mathcal{I}_t$ be the node set that is activated by the tth propagation step. In this case, we have $\mathcal{S}_0 = \mathcal{S}$ and $\mathcal{I}_0 = \mathcal{S}$. Moreover, we set $t \leftarrow 0$.

Propagation: We set $t \leftarrow t + 1$. For each node $v_i \in \mathcal{I}_{t-1}$ and each node $v_j \in \mathcal{T}_i$, the node v_j is activated with a propagation probability p_{ij}. More specifically, for each such a pair of v_i and v_j, if a randomly generated value r_{ij} is higher than p_{ij}, the node v_j is added into $\mathcal{I}_t$ (i.e., $s_j \leftarrow 1$). Then, we set $\mathcal{S}_t \leftarrow \mathcal{S}_{t-1} \cup \mathcal{I}_t$.

Termination: The propagation step is iteratively and independently executed until no further node is activated, i.e., $|\mathcal{I}_t| = 0$.

Note that, for a large-scale network, it is time-consuming to execute the above IC propagation processes as the final t at the termination step may be very large and it is hard to determine the final t value *a priori*. To simplify the propagation processes, Lee and Chung proposed a fast approximation IC model, in which the influence ability of seed nodes is limited within their 2-hop range (such as neighbors' neighbor). In this case, the final t value is no greater than 2. Fig. 1 gives a schematic illustration of the approximation IC model. Extensive studies and experiments have shown that the fast approximation IC model can effectively evaluate the influence spread of the IC model [32].

B. Problem Formulation for IM Under a Discrete Optimization Model

Given a complex network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ and the predefined size k of seed node set, the IM problem tries to find an optimal seed node set $\mathcal{S}$ with $|\mathcal{S}| = k$, so as to maximize the propagation of influence under the approximation IC model. This IM problem can be formulated as the following discrete optimization problem:

$$\begin{aligned} \max\ & \mathbf{F}(\mathcal{S}) \\ &= \sum_{v_i \in \mathcal{S}} \mathbf{F}(v_i) - \left(\sum_{v_i \in \mathcal{S}} \sum_{v_j \in \mathcal{T}_i \cap \mathcal{S}} p_{ij} \cdot (\tau(v_j) - p_{ji}) \right) \\ &\quad - \sum_{v_i \in \mathcal{S}} \sum_{v_j \in \mathcal{T}_i - \mathcal{S}} \sum_{v_k \in \mathcal{T}_j \cap \mathcal{S}} p_{ij} p_{jk} \end{aligned} \quad , \quad (1)$$

where $\mathbf{F}(v_i)$ and $\mathbf{F}(\mathcal{S})$ evaluates the number of nodes (the 2-hop influence spread) that can be influenced by the node v_i and node set $\mathcal{S}$ under the approximation IC model, respectively, and $\tau(v_j)$ denotes the 1-hop influence spread of node v_j that can be computed as $\tau(v_j) = 1 + \sum_{v_k : v_k \in \mathcal{T}_j} p_{kj}$. As shown in (1), $\mathbf{F}(\mathcal{S})$ is evaluated based on three parts. The first part evaluates the sum of influence spread of all nodes in $\mathcal{S}$, while the second and third parts compute the redundant 1-hop and 2-hop influence spread caused by different seed nodes, respectively. The $\mathbf{F}(\mathcal{S})$ value is in the range of k to n, and a high value corresponds to a good solution $\mathcal{S}$ to the IM problem.

In the optimization problem, $\mathcal{S}$ can be represented by a $n-$ bit binary variable $\mathbf{x}$ or a $k-$ bit real variable $\mathbf{y}$. More specifically, we let $\mathbf{x} = \{x_i \in \{0, 1\}\}_n$ denote the seed set of nodes $\mathcal{V}$ of $\mathcal{G}$, where $x_i = 1$ represents that the node v_i is the seed node; and $x_i = 0$ otherwise. We let $\mathbf{y} = \{y_i \in [1, 2, \ldots, n]\}_k, \forall i \neq j, y_i \neq y_j$ denote the seed set of nodes $\mathcal{V}$ of $\mathcal{G}$, where $y_i = a$ represents that the node v_a is the seed node. In this case, the optimization problem in (1) can be modeled as classical discrete combinational optimization problems with variables $\mathbf{x}$ or $\mathbf{y}$.

Note that, this discrete optimization problem is a NP-hard problem and has wide solution space (such as 2^n or n^k possible solutions). In this case, classical approximate or meta-heuristic IM algorithms [22]–[26] find it hard to obtain a good tradeoff between effectiveness and efficiency. To solve this issue, we first model the IM problem as a continuous parameter optimization of DQN (see Section III), and then propose an evolutionary DRL algorithm (EDRL-IM).

III. Our Work: The Continuous DQN Parameter Optimization Problem Model for the IM Problem

In this section, we design a problem-specific DQN for the selection of seed nodes for the IM problem, and transform the discrete optimization IM problem in (1) into a continuous DQN parameter optimization problem.

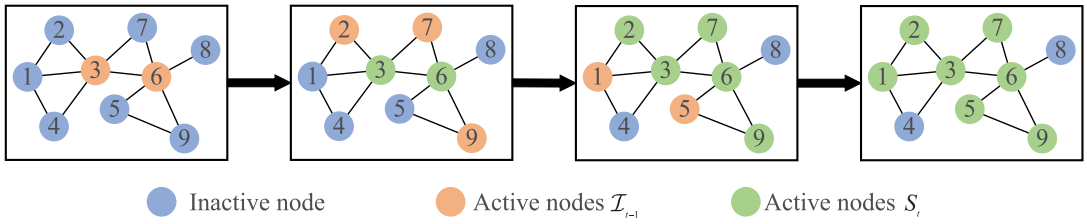


Fig. 1. A schematic illustration of the spreading processes of influence spreading under the approximation IC diffusion model. We let $\mathcal{S} = \mathcal{S}_0 = \mathcal{I}_0 = \{3, 6\}$ be the seed node set. Then, the nodes in $\mathcal{I}_0$ will activate their neighbors nodes $j \in \{1, 2, 4, 7, 5, 8, 9\}$ according to the propagation probability. Here, nodes $\mathcal{I}_1 = \{2, 7, 9\}$ were successfully activated, and $\mathcal{S}_1 = \mathcal{S}_0 \cup \mathcal{I}_1 = \{2, 3, 6, 7, 9\}$. Subsequently, the above activation steps are iteratively occurred until no further nodes can be activated. The final influenced nodes under $\mathcal{S}_0 = \{3, 6\}$ are nodes $\{1, 2, 3, 5, 6, 7, 9\}$.

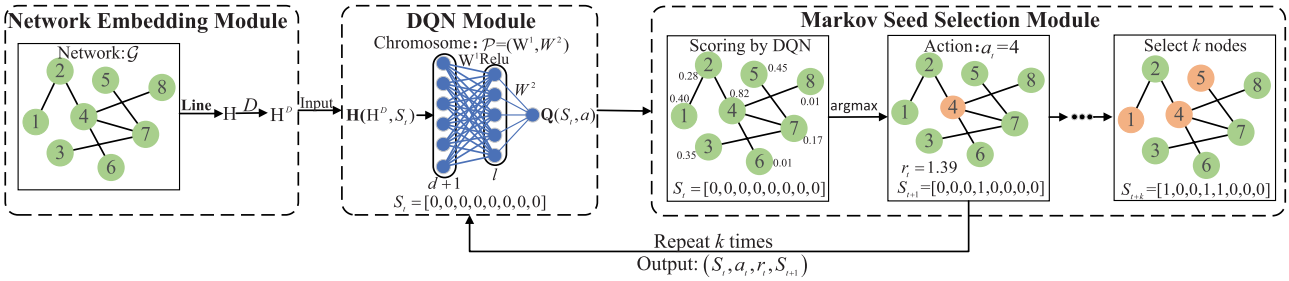


Fig. 2. Schematic illustration of the problem-specific DQN.

The problem-specific DQN (see Fig. 2) consists of three modules: network embedding, DQN and markov seed selection modules. The network embedding module learns the potential features of nodes; the DQN module gives node scores based on the node features; and the markov seed selection module uses the markov decision processes of a reinforcement learning (RL) and a DQN to choose seed nodes. With these modules, the IM problem in (1) can be well transformed into a continuous DQN parameter optimization problem.

A. Network Embedding Module

The network embedding module tries to embed a high-level network representation B into a low-dimensional node representation $\mathrm{H} = [h_{ik}]_{n \cdot d}$, $v_i \in \mathcal{V}$, which enables to extract the potential features of nodes and transform a discrete network representation into a continuous feature representation, where d is the predefined number of embedding dimension and $h_{ik} \in [-1, 1]$ is the k-th feature value of node v_i. These features can be well used for network tasks such as community detection, link prediction, influence propagation, and item recommendation.

In recent years, certain representative network embedding methods have been proposed to extract the low-dimensional features of nodes for networks, including DeepWalk [33], Node2vec [34], SDNE [35], and Line [36]. Here, we choose the Line method due to its low computational complexity, good scalability and good performance in preserving both the 1-hop and 2-hop node influence of networks.

The Line method first models the network embedding problem as an optimization of information similarity objective that preserves both the first and second node proximities of networks, and then adopts an asynchronous stochastic gradient algorithm [37] to optimize the objective. The first and second node proximities reflect the one-hop and two-hop node influence of networks, respectively. Subsequently, to tackle large-scale networks, the Line method proposes an edge sampling technique to improve effectiveness and efficiency. Finally, it returns the low-dimensional node representations H of networks. The details of the Line method could be found in [36].

Accordingly, given the Line method ($\mathbf{Line}()$), the network embedding module can be formulated as follows:

$$\mathrm{H} \leftarrow \mathbf{Line}(\mathrm{B}, d), \mathrm{H} = [h_{ik}]_{n \cdot d}, h_{ik} \in [-1, 1]. \tag{2}$$

Moreover, we let $\mathrm{H}^D \leftarrow [\mathrm{H}\ D]$ that contains the learned local link features H and the global node degree feature D of nodes.

B. DQN Module

The DQN module uses a three-layer fully connected DQN with size $(d+1)/l/1$ (see Fig. 2) to integrate the features H^D of nodes into a composite feature, aiming to approximate to the state-activation value function $\mathbf{Q}(S, a)$ in the RL, where $S = \{S_i \in [0\ 1]\}_n$ denotes the seed state of nodes while $a \in [1, n]$ represents the activation of choosing a seed node. Here, for each $i \in \{1, 2, \ldots, n\}$, $S_i = 1$ represents that node v_i is the seed node, and $S_i = 0$ denotes otherwise. Moreover, $a = j$ denotes that the node v_j is chosen as a seed node. More specifically, the state-activation value function $\mathbf{Q}(S, a)$ evaluates the possible reward when taking action a at state S, and it can be evaluated by the DQN as follows:

$$\begin{aligned} \mathbf{Q}(S, a) &= \mathbf{DQN}(\mathrm{W}^1, W^2, \mathrm{H}^D, S) \\ &= \mathbf{Relu}(\mathbf{H}(\mathrm{H}^D, S) \cdot \mathrm{W}^1) \cdot W^2, \end{aligned} \tag{3}$$

where the function $\mathbf{DQN}$ represents the DQN which generates an output $\mathbf{Relu}(\mathbf{H}(\mathrm{H}^D, S) \cdot \mathrm{W}^1) \cdot W^2$, while W^1 and W^2 are the weigh parameters of the DQN at the first and second layers,

respectively (see Fig. 2). $\mathbf{H}$ denotes a node feature chosen function. For each i, $\mathbf{H}(\mathrm{H}_{i\cdot}^{D}, S_i)$ is evaluated as follows:

$$\mathbf{H}(\mathrm{H}_{i\cdot}^{D}, S_i) = \mathrm{H}_{i\cdot}^{D} \cdot (1 - S_i).$$

The $\mathbf{H}$ function tries to eliminate the influence of seed nodes, which enables to choose the most influential nodes among the rest of non-active nodes. Moreover, $\mathbf{Relu}$ is an activation function which tries to eliminate negative feature values by setting the variables with negative values to zero.

C. Markov Seed Selection Module

The markov seed selection module aims to use the markov decision processes of RL and a DQN to choose seed nodes for the IM problem. Here, a markov decision process of RL is a discrete time decision process defined as a 4-touple (S_t, a_t, r_t, S_{t+1}), where S_t and S_{t+1} are the seed states of nodes at time t and $t+1$, respectively, while a_t and r_t are the seed node activation and the reward after a_t at time t, respectively.

To avoid the influence overlapping of seed nodes, we present a dynamic markov seed selection strategy to choose seed nodes (see **Algorithm** 1). This strategy consists of k discrete time decision processes, and each process first uses a DQN to compute the state-activation value function $\mathbf{Q}(S_t, a)$, and then adds the node with the highest value in $\mathbf{Q}(S_t, a)$ into the seed set. Moreover, each process will return a 4-touple (S_t, a_t, r_t, S_{t+1}) through a DQN. Formally, for each time t, the markov seed decision process is represented as follows:

$$\mathbf{Q}(S_t, a) = \mathbf{DQN}(\mathrm{W}^1, W^2, \mathrm{H}^D, S_t), \tag{4}$$

$$\mathbf{Q}(S_t, a_t) = \max_a \mathbf{Q}(S_t, a), \tag{5}$$

$$a_t = \arg\max_a \mathbf{Q}(S^t, a), \tag{6}$$

$$S_{t+1} = S_t,\ S_{t+1}(a_{t+1}) = 1, \tag{7}$$

$$r_t = \mathbf{F}(\mathbf{O}(S_{t+1})) - \mathbf{F}(\mathbf{O}(S_t)), \tag{8}$$

where the function $\mathbf{DQN}$ represents the DQN. Moreover, the function $\mathbf{O}(S)$ transforms a vector S to a seed node set $\mathcal{S}$. More specifically, $\mathbf{O}(S)$ is computed as follows:

$$\mathbf{O}(S) = \mathcal{S} = \{v_i \in \mathcal{V} : S_i = 1\}.$$

(4) computes the state-activation value of an agent under state S_t at time t; (5) evaluates the maximum state-activation value under all possible a at state S_t; (6) executes the activation a_t that chooses the node v_{a_t} with the highest state-activation value at time t as the seed node; (7) updates the seed state S_{t+1} at time $(t+1)$; and (8) computes the 2-hop influence objective reward when the activation a_t is executed.

D. Continuous DQN Parameter Optimization Model for the IM Problem

With the aforementioned models, we can use a DQN with weight parameters $[\mathrm{W}^1\ W^2]$ to choose seed nodes, and adopt the objective in (1) to evaluate their influence propagation. In this case, each possible DQN returns a seed set for the IM problem. To obtain an optimal seed set, we can model the IM problem as a continuous DQN weight parameter optimization problem which can be represented as follows:

$$\begin{aligned} \max\ & \mathbf{F}(\mathbf{O}(S_k)) = \mathbf{F}(\mathrm{W}^1, W^2) \\ \text{s.t. } & \mathbf{DQN}(\mathrm{W}^1, W^2, \mathrm{H}^D, S), \end{aligned} \tag{9}$$

where $S = [\,]$ is the initial seed node state at time $t = 0$, while S_k is the final seed node state computed by the aforementioned markov seed selection strategy.

Algorithm 1: Markov Seed Selection Strategy.

1: **Input**: A DQN with W^1 and W^2, size of seed set k, and the feature H^D
2: **Output**: Seed set $\mathcal{S}$ and a set of markov decision processes $\{(S_t, a_t, r_t, S_{t+1})\}_k$
3: Set $t \leftarrow 1$ and $S_t \leftarrow [\,]$.
4: **while** $(t \leq k)$ **do**
5: Compute $\mathbf{Q}(S_t, a)$ based on (4).
6: Compute a_t based on (5) and (6).
7: Compute S_{t+1} and r_t based on (7) and (8), respectively.
8: $t \leftarrow t + 1$.
9: **end while**

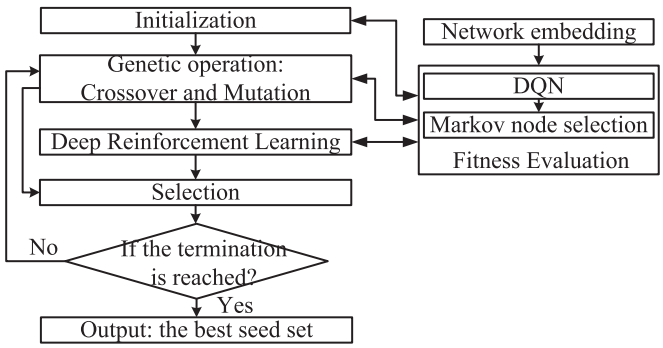


Fig. 3. The flowchart of EDRL-IM framework.

IV. Our Work: The Proposed EDRL-IM Algorithm for the IM Problem

In this section, an evolutionary deep reinforcement learning algorithm (called EDRL-IM) is proposed, aiming to find an optimal DQN for the IM problem in (1).

EDRL-IM combines an EA and a DRL to evolve the DQN. The EA simultaneously evolves a population of individuals, each of which represents a DQN and returns a solution to the IM problem through the aforementioned seed selection strategy, while the DRL integrates all information and network-specific knowledge of DQNs to accelerate their evolution.

The framework of EDRL-IM is shown in **Algorithm 2**, and the flowchart is shown in Fig. 3. More specifically, EDRL-IM first generates an initial population $\mathcal{P}(0)$ with size n_p by using the initialization (**Initialization**()), and then uses the evolutionary operations such as genetic operation (**Genetic_Operation**())

Algorithm 2: Framework of EDRL-IM.

Input: population size: n_p; parent population size: n_o; crossover probability: p_c; mutation probability: p_m; maximum number of generations: n_g.
$\mathcal{P}(0) \leftarrow$ **Initialization**(n_p).
for $(q = 1$ to $n_g)$ **do**
  $\mathcal{P}^s(q) \leftarrow$ **Genetic_Operation**$(\mathcal{P}(q), n_o, p_c, p_m)$, and find the solution $\mathcal{P}^f(q)$ with the maximum $\mathbf{F}$ in $\mathcal{P}^s(q)$.
  $\mathcal{P}^f(q) \leftarrow$ **DRL**$(\mathcal{P}^f(q))$.
  $\mathcal{P}(q+1) \leftarrow$ **Selection**$(n_p, \mathcal{P}(q), \mathcal{P}^s(q), \mathcal{P}^f(q))$.
end for

Algorithm 3: Evaluation().

Input: A solution $\mathcal{P}_i : \mathrm{W}_i^1, W_i^2, k$ and H^D.
Output: $\mathcal{S}$, $\mathbf{F}(\mathcal{S})$, and $\{(S_t^i, a_t^i, r_t^i, S_{t+1}^i)\}$.
Compute $\mathcal{S}, \{(S_t^i, a_t^i, r_t^i, S_{t+1}^i)\}$ based on **Algorithm 1**.
Compute $\mathbf{F}(\mathcal{S})$ based on (1).

and selection (**Selection()**) and the reinforcement learning operation (**DRL()**) to evolve the population. Generally, the evolutionary operations facilitate exploration, while the reinforcement learning operation facilitates exploitation. These operations are iteratively executed n_g times and finally output an optimal DQN and a seed node set for the IM problem, where n_g is the predefined number of iterations.

In the following subsections, the operations of EDRL-IM are introduced in detail.

A. Evolutionary Operation

1) Solution Representation and Evaluation: In EDRL-IM, a population $\mathcal{P} = \{\mathcal{P}_i\}_{n_p}$ of solutions are simultaneously evolved, where each solution $\mathcal{P}_i$ represents a DQN. Formally, $\mathcal{P}_i$ is represented as follows:

$$\mathcal{P}_i = (\mathrm{W}_i^1, W_i^2), \mathrm{W}_i^1 \in [-1, 1]_{(d+1)\cdot l}, W_i^2 \in [-1, 1]_{l\cdot 1}.$$

Recall that W^1 and W^2 are the weight parameters of the first and second layers of a DQN, respectively. $(d+1)$ and l are the number of neural nodes in the first and second layers, respectively.

For each solution $\mathcal{P}_i$, EDRL-IM combines the aforementioned network embedding, DQN and seed markov selection modules and the objectives in (1) to generate two outputs. One of the outputs is the seed node set $\mathcal{S}^i$ and its influence spread $\mathbf{F}^i$, which is used for the fitness evaluation of the solution $\mathcal{P}_i$ in the EA steps. The other one is a set of markov decision processes $\{(S_t^i, a_t^i, r_t^i, S_{t+1}^i)\}$ that are used for the acceleration of the DQN optimization in the DRL steps.

Algorithm 3 gives the detailed operations of a solution evaluation (Functioned as **Evaluation()**).

2) Initialization: The initialization (**Initialization()**) tries to generate a population of solutions. For our IM problem, there is no prior knowledge about the DQNs in the initialization stage. Therefore, to simplify the initialization operation, for each weight parameter of the DQNs, we randomly generate a value in the range of -1 to 1 as its initial value. Algorithm 4 gives the detailed operations of the population initialization.

Algorithm 4: Initialization().

Input: Size of population: n_p.
Output: Initial solutions $\mathcal{P}(0)$.
Set $\mathcal{P}_i(0) = (\mathrm{W}_i^1, W_i^2), i = 1, 2, \ldots, n_p$.
for $(i = 1$ to $n_p)$ **do**
  for (each element in W_i^1 and W_i^2) **do**
    Randomly generate a value in the range of -1 to 1.
  end for
  $[\mathcal{S}\ \mathbf{F}(\mathcal{S})\ \{(S_t^i, a_t^i, r_t^i, S_{t+1}^i)\}] \leftarrow$ **Evaluation**$(\mathcal{P}_i(0))$.
end for

Algorithm 5: Genetic_Operation().

Input: Parent solutions: $\mathcal{P}(q)$, crossover probability: p_c and mutation probability p_m.
Output: Offspring solutions: $\mathcal{P}^s(q)$.
Sort the solutions $\mathcal{P}(q)$ in a descend order based on their fitness.
Generate a set of pairs of parent solutions $\{(\mathcal{P}_i(q), \mathcal{P}_j(q))\}$, where $\mathcal{P}_i(q)$ and $\mathcal{P}_j(q)$ randomly comes from the top and last 50% solutions in $\mathcal{P}(q)$, respectively.
for (each pair of solutions $(\mathcal{P}_i(q), \mathcal{P}_j(q))$) **do**
  if (A randomly generated value $p \leq p_c$) **then**
    Randomly generate a bit value 0 or 1 for each gene position.
    Cross the gene values of the two solutions that have a bit value 1, and then generate two offspring solutions $\mathcal{P}_i^s(q)$ and $\mathcal{P}_j^s(q)$.
  else
    Put $\mathcal{P}_i(q)$ and $\mathcal{P}_j(q)$ into $\mathcal{P}^s(q)$.
  end if
end for // Crossover
for (Each solution $\mathcal{P}_i^s(q)$ in $\mathcal{P}^s(q)$) **do**
  for (Each gene position of $\mathcal{P}_i^s(q)$) **do**
    if (A randomly generated value $p \leq p_m$) **then**
      Update the gene value to a value $\mathbf{N}(-1, 1)$.
    end if
  end for
  $[\mathcal{S}\ \mathbf{F}(\mathcal{S})\ \{(S_t^i, a_t^i, r_t^i, S_{t+1}^i)\}] \leftarrow$ **Evaluation**$(\mathcal{P}_i^s(q))$.
end for // Mutation

3) Genetic Operation: The genetic operations include crossover and mutation, which aim to guide the exploration of the proposed EDRL-IM. The crossover explores offspring solutions $\mathcal{P}^s(q)$ with high quality by recombing and inheriting parent solutions $\mathcal{P}(q)$, while the mutation explores new solutions with good genetic diversity by randomly tuning the offspring solutions.

The crossover works on the parent solutions $\mathcal{P}(q)$, and it mainly contains the operators of solution pairing and single-point crossover. The solution pairing operation generates a set

Algorithm 6: Selection().

1: **Input**: Size of population: n_p, the parent population $\mathcal{P}(q)$, the offspring population $\mathcal{P}^s(q)$, and the solution $\mathcal{P}^f(q)$ generated by the DRL.
2: **Output**: The population of next generation $\mathcal{P}(q+1)$.
3: Sort the solutions in $\mathcal{P}(q)$, $\mathcal{P}^s(q)$, and $\mathcal{P}^f(q)$ in a descend order based on their fitness.
4: Select $n_p/2$ solutions with high fitness to $\mathcal{P}(q+1)$.
5: Randomly select $n_p/2$ solutions from $\mathcal{P}(q) \bigcup \mathcal{P}^s(q) \bigcup \mathcal{P}^f(q)$ -$\mathcal{P}(q+1)$ to $\mathcal{P}(q+1)$.

Algorithm 7: DRL().

1: **Input**: Size of batches n_b, number n_t of steps in DQN, and $\mathcal{P}^f(q)$.
2: **Output**: $\mathcal{P}^f(q)$.
3: Construct the experience replay buffer $\mathcal{B}$.
4: Randomly sample the batches $\mathcal{C}$ with size n_b from $\mathcal{B}$, in which each batch is composed of n_t+1 tuples.
5: Adopt a SGD method to minimize $\mathbf{L}(\mathrm{W}^1, W^2)$ in (10), where the weigh parameters (W^1, W^2) of the DQN solution $\mathcal{P}^f(q)$ are updated as (11) and (12) at each iteration.

of pairs of parent solutions, while the single-point crossover executes the recombing and inheriting process for each pair of parent solutions to generate offspring solutions. Here, to learn effectively the structures of solutions in the crossover, the solution pairing operation generates a set of pairs of parent solutions $\{(\mathcal{P}_i(q), \mathcal{P}_j(q))\}$, in which $\mathcal{P}_i(q)$ and $\mathcal{P}_j(q)$ randomly comes from the top 50% and the last 50% solutions, respectively. For each pair of solutions $(\mathcal{P}_i(q), \mathcal{P}_j(q))$, the single-point crossover first randomly generates a bit value 0 or 1 for each gene position, and then crosses the gene values of the two solutions that have a bit value 1.

The mutation works on the offspring solutions $\mathcal{P}^s(q)$. To improve the solution diversity, a normal random mutation is chosen, which works on each solution $\mathcal{P}_i^s(q)$ generated by the crossover. More specifically, for each gene position in $\mathcal{P}_i^s(q)$, if a random value is smaller than a mutation probability p_m, its gene value is mutated to a value $x = \mathbf{N}(-1, 1)$ in the range of [-1, 1] that follow a probability density function $e^{-x^2/2}/\sqrt{2\pi}$, where $\mathbf{N}(-1, 1)$ is the normal random mutation function.

Algorithm 5 gives the procedures of the proposed genetic operation.

4) Selection: The selection operation selects the population of the next generation from the parent population $\mathcal{P}(q)$, the offspring population $\mathcal{P}^s(q)$, and the solution $\mathcal{P}^f(q)$ generated by the DRL, which aims to preserve both the solution effectiveness and diversity of population. Here, inspired by the work of Khadka [38], we select half of the solutions with high fitness to the population of next generation to preserve the solution effectiveness and increase the stability of the population. Moreover, to preserve the diversity of population, we randomly select half of the rest of solutions to the population of next generation. **Algorithm 6** gives the procedures of this selection operation. Although the other selection strategies such as the roulette wheel selection can be also used here, their performance for the IM problem is worse than the selection strategy of the EDRL-IM in our simulation results.

B. Reinforcement Learning Operation

In EDRL-IM, the reinforcement learning operation ($\mathbf{DRL}()$, see **Algorithm** 7) is used for exploitation around the solution $\mathcal{P}^f(q)$ in $\mathcal{P}^s(q)$ that has the highest 2-hop influence spread. In order to obtain a good tradeoff between the effectiveness and efficiency in learning the parameters in $\mathcal{P}^f(q)$, we use a combination of a n_t-step Q-learning [31] and fitted Q-iteration [31]. The n_t-step Q-learning considers the delayed rewards of actions, where the final estimated reward $\mathbf{Q}(S_t, a_t)$ of an action a_t at state S_t is only received after an episode with n_t-step exact feedbacks. The fitted Q-iteration adopts batch samples $\mathcal{C}$ of experience replay buffer $\mathcal{B}$ (markov decision) samples returned by evaluation steps to update the function approximation. In this case, the objective of RL is to minimize the n_t-step loss function $\mathbf{L}$ of a DQN with weight parameters (W^1, W^2), which evaluates the expected loss between the estimated reward $\mathbf{Q}(S_t, a_t)$ returned by a DQN and the approximated reward $\widetilde{\mathbf{Q}}(S_t, a_t)$ estimated by the n_t-step exact reward feedbacks and the estimated reward $\mathbf{Q}(S_{t+n_t}, a_{t+n_t})$ in the batch samples $\mathcal{C}$. More specifically, the n_t-step loss function $\mathbf{L}(\mathrm{W}^1, W^2)$ is evaluated as follows:

$$\mathbf{L}(\mathrm{W}^1, W^2) = \mathbf{E}_{\mathcal{C}\in\mathcal{B}}\left[\left(\sum_{i=0}^{n_t-1} r_{t+i} + \gamma \max_{a'} \mathbf{Q}(S_{t+n_t}, a'; \mathrm{W}^1, W^2) - \mathbf{Q}(S_t, a_t; \mathrm{W}^1, W^2)\right)^2\right], \tag{10}$$

where γ is a decaying factor which determines the importance of future rewards, while $\mathbf{E}$ is an expectation function.

To minimize $\mathbf{L}(\mathrm{W}^1, W^2)$, the experience replay buffer $\mathcal{B}$ and the batch samples $\mathcal{C}$ are first constructed. $\mathcal{B}$ is collected from the evolutionary processes while $\mathcal{C}$ is randomly sampled from $\mathcal{B}$. Then, the $\mathbf{L}(\mathrm{W}^1, W^2)$ is minimized by a stochastic gradient descent (SGD) method, where the weight parameters (W^1, W^2) are updated as follows:

$$(\mathrm{W}^1, W^2)' = (\mathrm{W}^1, W^2) - \Delta(\mathrm{W}^1, W^2), \tag{11}$$

where $\Delta(\mathrm{W}^1, W^2)$ is computed as follows:

$$\Delta(\mathrm{W}^1, W^2) = \alpha \cdot \left[\sum_{i=0}^{n_t-1} r_{t+i} + \gamma \max_{a'} \mathbf{Q}(S_{t+n_t}, a'; \mathrm{W}^1, W^2) - \mathbf{Q}(s_t, a_t; \mathrm{W}^1, W^2)\right] \nabla_{(\mathrm{W}^1, W^2)} \mathbf{Q}(S_t, a_t; \mathrm{W}^1, W^2), \tag{12}$$

where α is a learning rate, while $\nabla_{(\mathrm{W}^1, W^2)}$ is the deviation to (W^1, W^2).

TABLE I
DETAILED INFORMATION OF THE TESTED REAL-WORLD NETWORKS

Network	n	m	$\bar{d}$	Basic description
Polbooks [41]	105	882	16.80	It represents the purchase of books about U.S. politics in Amazon.
SFI [42]	118	200	3.125	It represents the collaboration of scientists who lived at the Santa Fe Institute between 1999 and 2000.
Jazz [43]	198	2,742	27.70	It represents the cooperation between jazz musicians. together.
Filmtrust [44]	874	1,853	4.240	It denotes the user trusts in the FilmTrust project.
Email [45]	1,133	5,451	9.622	It denotes the E-mail communication in Rovira i Virgili University.
Atc [46]	1,226	2,615	4.266	It denotes the route recommendation of USA's National Flight Data Center.
Power [47]	4,941	6,594	2.669	It shows the power topology of the western states of the United States.
Wiki [48]	7,115	103,689	29.15	It represents the user relationships in the Wiki website.
Gemo [41]	7,743	11,898	3.073	It represents the author's cooperation in the computational geometry.
Pgp [49]	10,680	24,340	4.558	It denotes the private communication between users extracted by the Pretty-Good-Privacy algorithm.

$\bar{d}$ Represents the Averaged Node Degree of a Network.

V. EXPERIMENTAL RESULTS

In this section, we test EDRL-IM and seven classical IM algorithms on the GN [39] and LFR [40] benchmark networks and ten real-world networks to validate the effectiveness of the proposed EDRL-IM. In the following, the details of experimental settings are first given, and then a comparison of EDRL-IM with the seven IM algorithms on the tested networks is made and the comparison results are analyzed.

A. Experimental Settings

1) Experimental Networks: To validate the effectiveness of EDRL-IM, we test it on the GN [39] and LFR [40] benchmark networks and ten real-world networks.

GN Benchmark Networks: The GN benchmark networks [39] are used to represent real modular networks, which have the same degree 16 for all 128 nodes. These nodes are evenly distributed into 4 communities, and each of them has a fraction $(1-\mu)$ of links that are located in a community, where μ is a control parameter. With the increase of μ, the number of links across communities is increased, and the GN benchmark network becomes more complex. Here, we test all algorithms on two GN benchmark networks with $\mu = 0.1$ and $\mu = 0.3$ to analyze their performance in tackling networks with homogeneous node degree.

LFR Benchmark Networks: The LFR benchmark networks [40] are extensions of the GN benchmark networks, which take the heterogeneity of node degree and community size into consideration. Moreover, they enable to model real-world networks with arbitrary network sizes. In these networks, the node degree and community size follow power law distribution with control parameters β and τ, respectively. Here, we test all algorithms on two LFR benchmark network with $\mu = 0.1$ and $\mu = 0.3$ to show their performance in tackling networks with heterogeneous node degrees, in which each network is set as $n = 500$, $\beta = 1$ and $\tau = 2$.

Real-World Networks: Ten real-world networks are tested to validate the effectiveness of EDRL-IM in tackling various real systems such as communication, cooperation, email, airline and social systems. Most of these real networks have clear community structures (see [50]). The detailed information of these networks is given in Table I.

2) Baseline Algorithms: Seven classical IM algorithms are chosen as baseline methods, including three approximate greedy algorithms (Degree, CELF [51] and CI [5]), two DRL algorithms (S2V-DQN [28] and Finder [31]) and two meta-heuristic algorithms (MA-IM$_{\text{multi}}$[25] and CMA-IM [7]). A comparison of EDRL-IM with the approximate greedy and meta-heuristic algorithms is made to show the effectiveness and efficiency of EDRL-IM in solving the IM problem, while a comparison of EDRL-IM with the DRL algorithms tries to validate the superiority of the evolutionary DRL framework over classical DRL frameworks.

Degree [7]: Degree first sorts the nodes of a network in a descend order based on their degree, and then selects the top k nodes with high node degree as a seed node set.

CELF [51]: CELF is an extension of Degree by taking the influence spread of nodes into consideration. It first calculates the influence spread range of each node, and then selects the node with the highest influence spread as a seed node. Next, it updates the influence spread of the nodes in the submodularities that contain seed nodes. The aforementioned steps are iteratively executed until all seed nodes in the seed set are selected.

CI [5]: CI selects the seed set by choosing a set of seed nodes that have the highest collective influence. For each node v_i, its collective influence ($I_C(v_i)$) is evaluated as follows:

$$I_C(v_i) = (\bar{d}_i - 1) \sum_{v_j \in Ball(v_i,\eta)} (\bar{d}_j - 1), \tag{13}$$

where $\bar{d}_i$ represent the degree of v_i while $Ball(v_i, \eta)$ is a collection of nodes with a distance of η from node v_i.

S2V-DQN [28]: S2V-DQN first uses the structure2vec method to embed a network into low-dimensional node representations, and then adopts a DRL with $\epsilon-$degree samples to train a DQN. Finally, the seed set is generated based on the seed scores generated by the DQN.

Finder [31]: Finder first uses a graph neural network for network embedding, and then uses a virtual node to aggregate the embedding results. Next, it trains a DQN with $\epsilon-$degree samples to give influence scores (the influence in the largest connected component) for all nodes. Finally, a set of nodes with high scores are chosen as seed nodes.

CMA-IM [7]: CMA-IM first models the IM problem as an optimization of 2-hop influence spread over communities, and then presents a memetic algorithm with a genetic algorithm and a local search to find seed nodes over communities. In CMA-IM, a population of initial seed sets are generated based on the node degree and community information of networks.

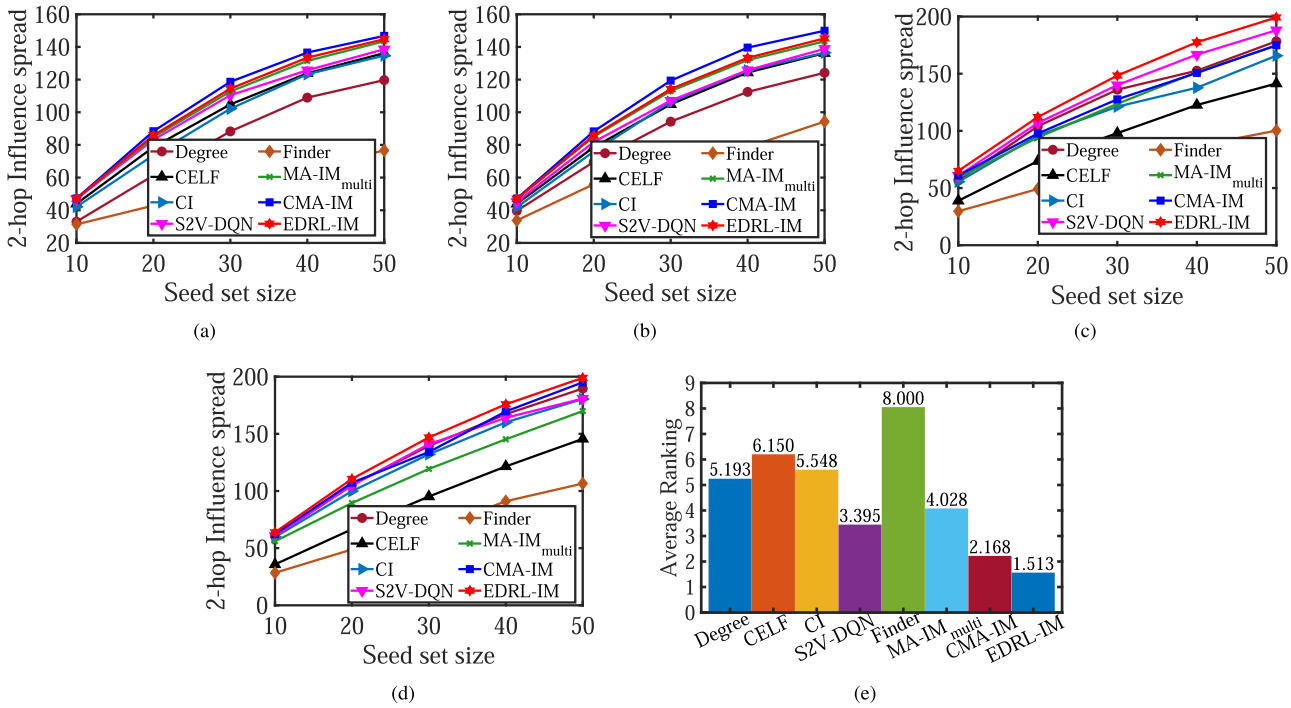


Fig. 4. Statistical performance of all algorithms on the GN and LFR benchmark networks under different seed set sizes. (a) and (b) are the 2-hop influence spread $\mathbf{F}$ of all algorithms on the GN benchmark networks with $\mu = 0.1$ and $\mu = 0.3$, respectively. (c) and (d) are the $\mathbf{F}$ results of all algorithms on LFR benchmark networks with $\mu = 0.1$ and (d) $\mu = 0.3$, respectively. (e) is the average performance ranking of all algorithms over all benchmark networks.

MA-IM$_{multi}$ [25]: MA-IM$_{\text{multi}}$ is an extension of CMA-IM, which enables to tackle multiplex networks. It first models the IM problem as an optimization of multiplex 2-hop influence spread, and then presents a novel MA algorithm to solve the IM problem in multiplex networks. In MA-IM$_{\text{multi}}$, a population of initial seed sets are generated based on an integration of random, roulette-based degree and distance selection strategies.

3) Criteria: To verify the performance of all algorithms, the 2-hop influence spread $\mathbf{F}(S)$ and the wilcoxon's rank sum test $+/-/\sim$ are adopted. The $\mathbf{F}$ evaluates the overall performance of all algorithms in finding seed nodes, while the $+/-/\sim$ compute the statistical difference of solutions obtained by all algorithms with a significance level 5%. The labels '+' and '-' indicate that the baseline algorithms have a better and worse performance than EDRL-IM, respectively, while the label '$\sim$' denotes that there is no significant performance difference between the baseline algorithms and EDRL-IM. Moreover, to validate the overall performance of all algorithms, we record their average performance ranking over all tested benchmark and real-world networks. More specifically, all algorithms are ranked from 1 to 8 based on their performance, and the algorithm with the best performance is ranked as 1. The overall performance ranking of an algorithm is its averaged performance ranking over all tested networks.

4) Simulation Settings: EDRL-IM is simulated by Python on a server with Intel (R), Xeon (R), E5-2630 v4 CPU (2.20 GHz) and 512 GB of RAM. Degree, CI, S2V-DQN and Finder are coded with python, while CELF and MA-IM$_{\text{multi}}$ are coded with C++. Moreover, CMA-IM is coded with matlab. For each network, all algorithms are independently tested for 20 trials with parameter settings in Table II. Moreover, all algorithms are evaluated under the propagation probability setting $p_{ij} = 0.1$ for each $e_{ij} \in \mathcal{E}$.

B. Experiments on the GN and LFR Benchmark Networks

In this part, all algorithms are tested on two GN and two LFR benchmark networks under five different settings of seed set size $k = \{10, 20, 30, 40, 50\}$, and the statistic $\mathbf{F}$ and overall performance rank results over 20 trials are recorded in Fig. 4.

As observed from Fig. 4, among the 20 influence test instances on the GN and LFR benchmark networks, EDRL-IM can obtain the highest influence spread $\mathbf{F}$ values on 14 test instances, while the other baseline IM algorithms can only obtain the highest $\mathbf{F}$ values on at most 6 instances. This shows the superior performance of EDRL-IM in finding influential seed nodes over the other baseline IM algorithms. The good performance of EDRL-IM is further validated by the overall performance ranks of all algorithms in Fig. 4(e). The results show that EDRL-IM has an average performance ranking (1.513) much lower than that of Degree (5.192), CELF (6.150), CI (5.548), S2V-DQN (3.395), Finder (8.000), MA-IM$_{\text{multi}}$ (4.028), and CMA-IM (2.168).

The results in Fig. 4 also show that the metaheuristic IM algorithms (EDRL-IM, CMA-IM and MA-IM$_{\text{multi}}$) have higher $\mathbf{F}$ and lower average performance ranking values than the approximate greedy algorithms (Degree, CELF and CI), especially for the comparison results in the GN benchmark networks. This is reasonable as these approximate greedy algorithms facilitate exploitation, but they suffer from the lack of exploration. Moreover, these meta-heuristic IM algorithms adopt a framework of

TABLE II
PARAMETER SETTINGS OF ALGORITHM

Algorithm	Parameter	Meaning	Value
EDRL-IM	n_p	Size of population	200
	p_c	Probability of crossover	0.8
	p_m	Probability of mutation	0.2
	n_g	Number of generations	100
	d	Dimension of network embedding vector	64
	l	Second layer size of DQN	64
	n_t	Number of DQN steps	4
	n_b	Batch sizes	512
	γ	Decaying factor	0.8
	α	Learning rate	0.001
CI	η	Node distance	1
S2V-DQN	hid_dim	Hidden layer dimension	64
	n_e	Number of eposides	50
	n_t	Number of DQN steps	4
	n_b	Batch size	64
	γ	Decaying factor	0.99
	α	Learning rate	0.001
Finder	d	Dimension of network embedding vector	64
	n_t	Number of DQN steps	4
	n_b	Batch size	64
	γ	Decaying factor	0.99
	α	Learning rate	0.0001
MA-IM$_{\text{multi}}$	n_p	Size of population	200
	p_c	Probability of crossover	0.6
	p_m	Probability of mutation	0.5
	p_l	Local search Probability	0.7
	n_g	Number of generations	100
CMA-IM	n_p	The size of population	200
	p_c	Probability of crossover	0.8
	p_m	Probability of mutation	0.2
	n_g	Number of generations	100
	α	Constant term	4
	β	Amplification term	10
	$pool$	Size of mating pool	100
	$tour$	Tournment size	2

MAs with GAs and local searches, in which the GAs facilitate exploration while the local searches are suitable for exploitation. Among these meta-heuristic IM algorithms, CMA-IM and EDRL-IM show the best performance in the GN and LFR benchmark networks, respectively. In the GN benchmark networks, EDRL-IM has a slightly worse performance than CMA-IM. This is because the GN benchmark networks have clear community structures and CMA-IM can use these community information to help finding seed nodes. Moreover, the GN benchmark networks have no obvious heterogeneous structure information. In this case, EDRL-IM is hard to learn useful structure information to improve its search and learning ability from the networks.

From Fig. 4, we can see that the approximate greedy algorithms (Degree, CELF and CI) have a good performance for the LFR benchmark networks with heterogeneous node degree and community structures, whereas they have a poor performance for the GN benchmarks networks. This is reasonable as the main greedy principles of these algorithms are designed based on the heterogeneous information of networks. We can also see that Degree, CELF and CI are easier to get trapped into local optima with the increase of the seed set sizes. This is because the solution search space for the IM problem is increased with the seed set sizes whereas these greedy algorithms are lack of exploration.

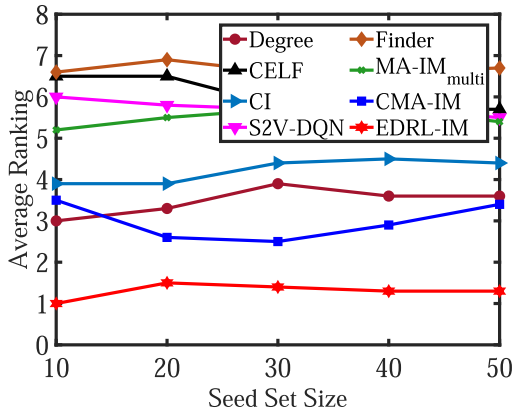


Fig. 5. Performance ranking of all algorithms with different seed set sizes on the real-world networks.

Fig. 4 also shows that in the DRL algorithms, S2V-DQN has a competitive performance whereas Finder has a poor performance for the IM problems in the GN and LFR benchmark networks. This validates that the DRL algorithms can effectively solve the IM problem, whereas they easily get trapped into local optima. This is because the DRL algorithms mainly adopt a greedy SGD optimization technique to optimize a solution and they use a greedy strategy with the highest reward value to choose seed nodes. Accordingly, their performance is highly related to the SGD optimization technique and the greedy strategies. To solve the issues of the DRL algorithms, we propose an evolutionary DRL framework (EDRL-IM), which combines an EA and a DRL to evolve a population of solutions simultaneously. In EDRL-IM, the EA is used for exploration while the DRL is adopted for exploitation. The combination of EA and DRL enables EDRL-IM to improve the search and learning capabilities of EA and DRL in solving the IM problem, which can be validated by the comparison results in Fig. 4.

C. Experiments on the Real-World Networks

To validate the practicability of EDRL-IM, we test all algorithms on the aforementioned real-world networks with different settings of seed set size $k = \{10, 20, 30, 40, 50\}$, and record their statistical results ($\mathbf{F}$, wilcoxons rank sum test and overall performance ranking) into Table III and Fig. 5. For each network, the best result among these algorithms is marked in boldface.

The results in Table III show that EDRL-IM is able to obtain the highest influence spread $\mathbf{F}$ values in 35 out of all 50 statistical values (70%), while Degree, CELF, CI, S2V-DQN, Finder, MA-IM$_{\text{multi}}$, and CMA-IM perform best in 1, 0, 2, 0, 0, 1 and 13 cases, respectively. Moreover, considering the comparisons with Degree, CELF, CI, S2V-DQN, Finder, MA-IM$_{\text{multi}}$, and CMA-IM, EDRL-IM shows an absolute advantage for the wilcoxons rank sum test, as it outperforms them on at least 35 cases, and only has a slightly worse performance than CMA-IM in 13 cases. In addition, considering the overall performance (see Fig. 5), EDRL-IM shows an obvious advantage over the other baseline algorithms. More specifically, EDRL-IM has an average performance ranking (1.310) much lower than that of

TABLE III
RESULTS OF ALL ALGORITHMS ON THE REAL NETWORKS WITH DIFFERENT SEED SET SIZES

Network	Seed set size	Degree	CELF	CI	S2V-DQN	Finder	MA-IM$_{\text{multi}}$	CMA-IM	EDRL-IM
Polbooks	k=10	36.22(-)	26.84(-)	35.01(-)	36.22(-)	25.47(-)	35.69(-)	35.17(-)	**37.06**
	k=20	47.56(-)	44.83(-)	46.36(-)	48.15(-)	46.20(-)	52.31(-)	51.92(-)	**53.08**
	k=30	54.29(-)	59.80(-)	51.95(-)	57.78(-)	56.81(-)	**66.14**(∼)	57.68(-)	66.02
	k=40	59.37(-)	71.50(-)	57.65(-)	70.90(-)	64.90(-)	78.66(-)	–(-)	**79.47**
	k=50	64.27(-)	80.08(-)	63.03(-)	78.50(-)	71.90(-)	89.07(-)	–(-)	**92.17**
Sfi	k=10	21.06(-)	18.40(-)	18.06(-)	16.64(-)	20.25(-)	20.56(-)	21.26(-)	**21.33**
	k=20	32.74(-)	29.25(-)	28.85(-)	28.19(-)	32.18(-)	31.30(-)	32.71(-)	**33.22**
	k=30	42.78(-)	38.99(-)	38.38(-)	38.89(-)	41.80(-)	41.27(-)	42.86(-)	**43.28**
	k=40	51.74(-)	48.89(-)	47.63(-)	49.21(-)	50.56(-)	50.98(-)	52.34(-)	**52.91**
	k=50	60.61(-)	58.48(-)	57.04(-)	58.54(-)	59.74(-)	60.38(-)	61.47(-)	**62.21**
Jazz	k=10	107.2(-)	80.33(-)	115.2(-)	63.71(-)	57.28(-)	109.6(-)	107.9(-)	**116.5**
	k=20	172.1(-)	134.5(-)	**181.2**(∼)	112.8(-)	82.91(-)	165.3(-)	162.6(-)	179.9
	k=30	207.6(-)	170.0(-)	223.4(-)	146.6(-)	99.50(-)	211.0(-)	211.8(-)	**227.1**
	k=40	230.3(-)	204.7(-)	249.2(-)	186.1(-)	115.3(-)	245.5(-)	251.0(-)	**263.7**
	k=50	253.5(-)	238.7(-)	270.6(-)	222.5(-)	131.7(-)	269.1(-)	–(-)	**285.7**
Filmtrust	k=10	35.28(-)	28.05(-)	32.60(-)	35.72(-)	36.65(-)	34.07(-)	39.32(-)	**41.41**
	k=20	51.90(-)	47.75(-)	49.81(-)	48.13(-)	56.97(-)	48.43(-)	**63.29**(+)	62.50
	k=30	71.02(-)	63.32(-)	65.02(-)	63.20(-)	75.41(-)	61.99(-)	**83.93**(+)	77.23
	k=40	82.56(-)	76.79(-)	75.26(-)	69.29(-)	85.66(-)	75.12(-)	**101.7**(+)	90.93
	k=50	91.08(-)	89.99(-)	84.45(-)	85.78(-)	97.66(-)	88.36(-)	**117.5**(+)	108.1
Email	k=10	131.3(-)	50.54(-)	131.8(-)	128.0(-)	21.61(-)	127.4(-)	134.2(∼)	**134.5**
	k=20	211.3(-)	102.3(-)	209.6(-)	194.5(-)	42.52(-)	192.9(-)	**220.0**(+)	219.6
	k=30	272.5(-)	147.9(-)	267.2(-)	273.7(-)	65.54(-)	244.1(-)	**286.6**(+)	283.1
	k=40	320.9(-)	183.3(-)	313.2(-)	325.0(-)	82.91(-)	283.8(-)	**340.0**(+)	335.2
	k=50	363.9(-)	220.7(-)	346.0(-)	300.0(-)	169.0(-)	319.4(-)	**381.2**(+)	375.0
Atc	k=10	34.42(-)	20.45(-)	32.98(-)	27.27(-)	33.41(-)	27.35(-)	34.27(-)	**34.89**
	k=20	57.15(-)	38.67(-)	51.73(-)	43.22(-)	55.07(-)	41.73(-)	**59.66**(+)	59.38
	k=30	77.11(-)	54.79(-)	72.27(-)	60.49(-)	73.89(-)	55.72(-)	**80.14**(+)	79.88
	k=40	93.46(-)	70.45(-)	87.46(-)	75.57(-)	89.61(-)	69.13(-)	**98.42**(+)	97.50
	k=50	106.5(-)	85.98(-)	102.0(-)	89.47(-)	106.1(-)	82.58(-)	**114.9**(+)	113.4
Power	k=10	20.57(-)	14.22(-)	18.35(-)	12.09(-)	14.67(-)	15.76(-)	20.11(-)	**20.66**
	k=20	37.95(-)	27.94(-)	35.43(-)	23.84(-)	28.39(-)	28.32(-)	37.99(-)	**38.22**
	k=30	54.68(-)	41.21(-)	50.36(-)	35.51(-)	41.37(-)	40.57(-)	54.69(-)	**54.97**
	k=40	70.47(-)	54.25(-)	65.55(-)	47.89(-)	55.04(-)	52.66(-)	70.85(-)	**70.99**
	k=50	85.82(-)	67.51(-)	79.56(-)	58.02(-)	68.54(-)	64.78(-)	86.69(∼)	**86.72**
Wiki	k=10	**2,872**(∼)	293.2(-)	**2,872**(∼)	–(-)	230.8(-)	1,387(-)	2,812(-)	**2,872**
	k=20	4,247(-)	559.5(-)	4,281(-)	–(-)	329.4(-)	1,755(-)	**4,371**(+)	4,348
	k=30	5,454(-)	764.5(-)	5,471(-)	–(-)	593.7(-)	1,955(-)	5,528(-)	**5,558**
	k=40	6,311(-)	874.7(-)	6,251(-)	–(-)	697.5(-)	2,263(-)	6,060(-)	**6,506**
	k=50	7,026(-)	1,017(-)	7,100(-)	–(-)	1,039(-)	2,548(-)	6,112(-)	**7,343**
Gemo	k=10	132.3(-)	102.0(-)	139.4(-)	–(-)	114.1(-)	80.17(-)	86.09(-)	**140.7**
	k=20	188.8(-)	154.5(-)	200.5(∼)	–(-)	143.7(-)	99.96(-)	149.8(-)	**200.7**
	k=30	229.4(-)	193.8(-)	236.8(-)	–(-)	179.9(-)	115.8(-)	205.8(-)	**244.3**
	k=40	258.5(-)	228.1(-)	264.6(-)	–(-)	195.9(-)	133.6(-)	252.2(-)	**276.0**
	k=50	282.4(-)	253.3(-)	286.5(-)	–(-)	227.1(-)	150.5(-)	285.8(-)	**303.9**
Pgp	k=10	166.5(-)	109.5(-)	171.8(-)	–(-)	78.86(-)	91.08(-)	156.4(-)	**173.6**
	k=20	259.5(-)	172.4(-)	264.7(-)	–(-)	133.9(-)	110.1(-)	245.9(-)	**269.2**
	k=30	332.7(-)	222.8(-)	345.5(-)	–(-)	183.1(-)	127.9(-)	326.8(-)	**349.3**
	k=40	398.9(-)	266.8(-)	403.7(-)	–(-)	224.4(-)	140.7(-)	385.7(-)	**416.0**
	k=50	448.3(-)	305.2(-)	463.1(-)	–(-)	254.9(-)	156.0(-)	433.9(-)	**470.4**
best/all	–	1/50	0/50	2/50	0/50	0/50	1/50	13/50	**35**/50
+/-/∼	–	0/49/1	0/50/0	0/47/3	0/50/0	0/50/0	0/49/1	13/35/2	–

'+,' '−' and '∼' Indicate That the Results of the Compared Algorithms are Significantly Better Than, Worse Than, and Similar to That of EDRL Respectively by Wilcoxon's Rank Sum Test With a Significance Level 5%. The Best Values are Highlighted in Bold Face.

Degree (3.481), CELF (6.072), CI (4.269), S2V-DQN (5.702), Finder (6.613), MA-IM$_{\text{multi}}$ (5.465) and CMA-IM (2.976). These comparison results validate the superiority of EDRL-IM over the other baseline methods in finding influential seed nodes.

Table III also presents some similar results as Fig. 4. For the small and medium scale networks (i.e., $n \leq 5,000$), the meta-heuristic IM algorithms (CMA-IM and EDRL-IM) have a better performance than the approximate greedy algorithms (Degree, CELF and CI). Moreover, the DRL algorithms (S2V-DQN and Finder) show a poor performance for the IM problem in networks. In addition, by combining the EA and DRL, our EDRL-IM shows a good performance for the IM problem in all networks. This validates the effectiveness of our proposed evolutionary DRL framework for solving the IM problem in networks.

Table III also shows that the approximate greedy algorithms (Degree, CELF and CI) have a competitive performance for most networks, especially for the large-scale networks (Gemo and Pgp). This is reasonable as compared with the benchmark networks, the real-world networks show more heterogeneous structure information. Moreover, most of the real networks show the scale-free property of node degree, which enables to

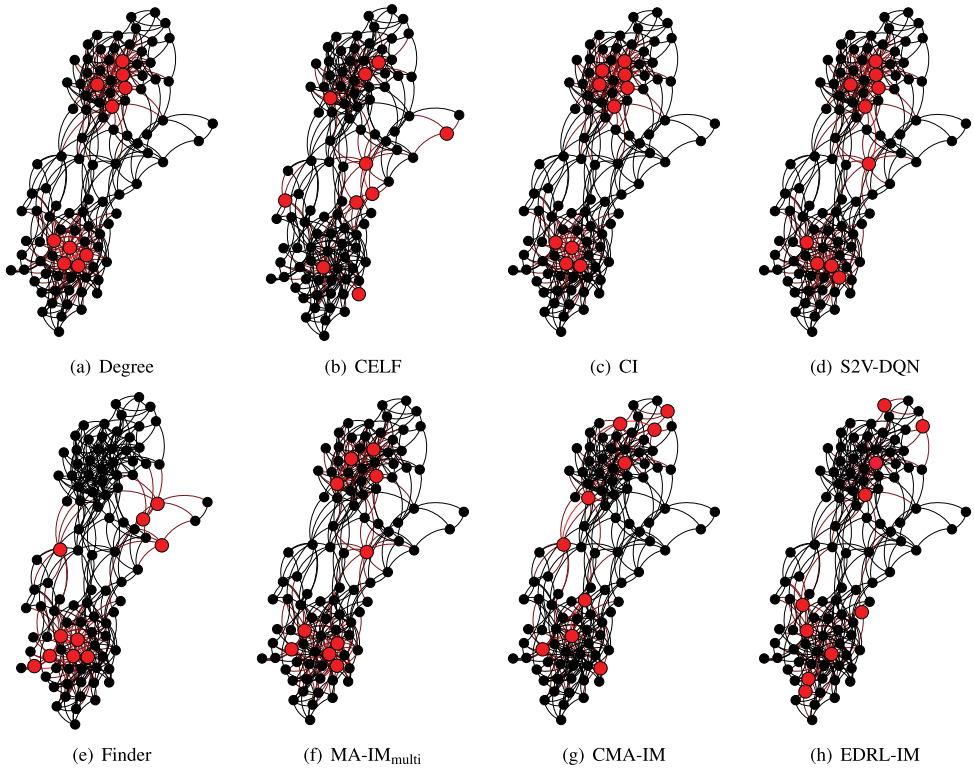


Fig. 6. Visualization of the seed node sets with size 10 found by all algorithms on the Polbooks network. The seed nodes are marked in red color.

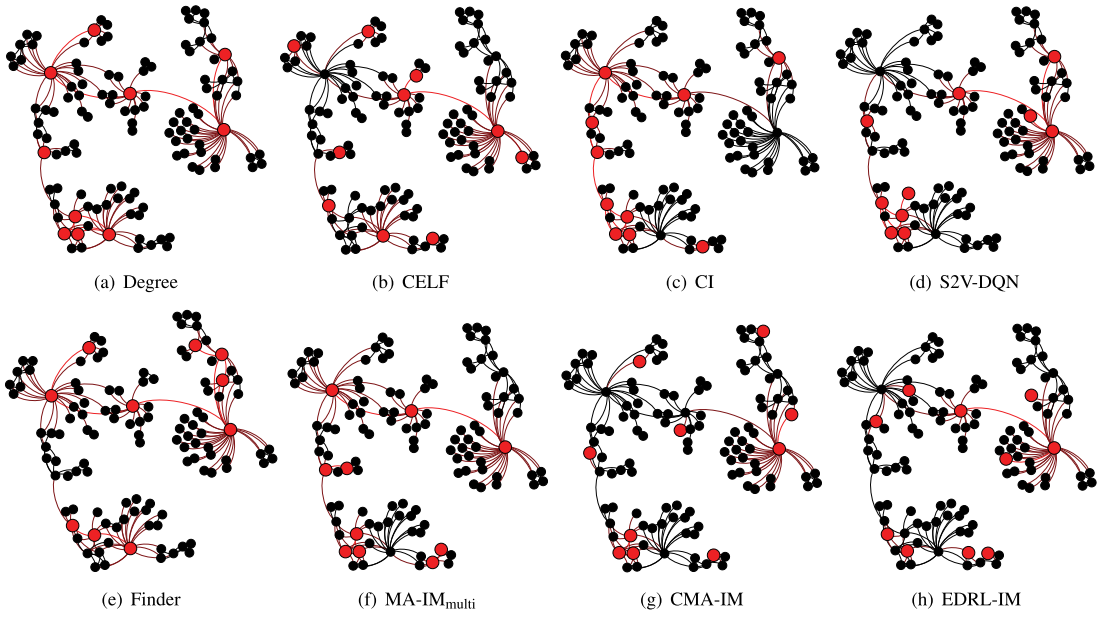
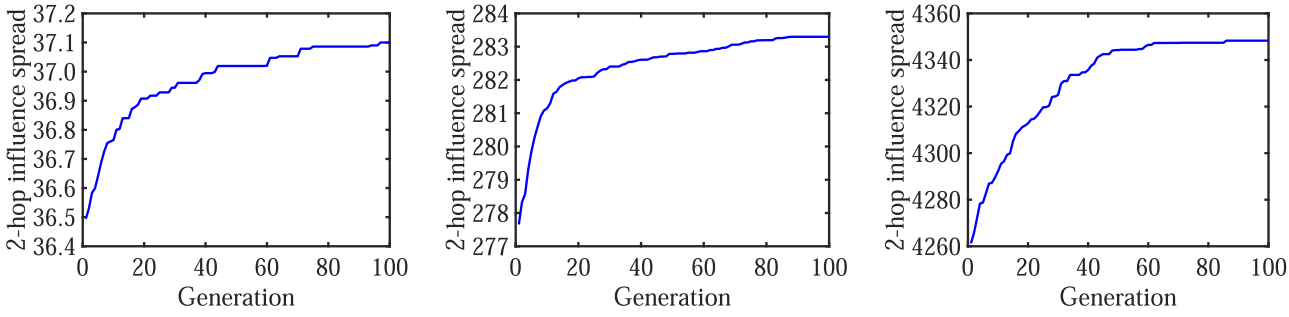


Fig. 7. Visualization of the seed node sets with size 10 found by all algorithms on the Sfi network. The seed nodes are marked in red color.

maximally spread influence through a few of nodes with high node degree.

The results in Table III present that with the increase of seed set size, the performance of approximate greedy algorithms (Degree, CELF and CI) and DRL algorithms (S2V-DQN Finder) are degraded. This is because the solution search space increases with the increase of the seed set size whereas the greedy and DRL algorithms are lack of exploration. The metaheuristic MA algorithms (MA-IM$_{multi}$, CMA-IM and EDRL-IM) combine the EA and local searches, which enable to facilitate exploration and exploitation, respectively. Note that, MA-IM$_{multi}$ and CMA-IM cannot work well for the large-scale networks (Gemo and Pgp) due to their limited search ability in the large-scale discrete solution search space.



(a) Polbooks-10 (b) Email-30 (c) Wiki-20

Fig. 8. 2-hop influence spread range of EDRL-IM on the Polbook ($k = 10$), Email ($k = 30$) and Wiki ($k = 20$) networks varies with the number of iterations.

To show the performance of EDRL-IM, we plot the visualization of the seed node set of the Polbooks and Sfi network found by all algorithms in Figs. 6 and 7, respectively. From Figs. 6 and 7, we can see that the seed sets selected by the Degree and CI algorithms are concentrated on nodes with high node degree. The seed sets selected by the CELF, S2V-DQN, Finder and MA-IM$_{\text{multi}}$ algorithms are relatively scattered, and most of them are the ones with high node degree. Therefore, Degree, CI, CELF, S2V-DQN, Finder and MA-IM$_{\text{multi}}$ easily get trapped into local optima. The seed nodes selected by the CMA-IM and EDRL-IM algorithms are scattered, and contain some low-degree nodes that connect different connected components. This enables CMA-IM and EDRL-IM to find more influential seed nodes. The main difference of the seed nodes found by the CMA-IM and EDRL-IM is that the seed nodes found by CMA-IM are evenly distributed over all communities. In this case, the influence of nodes within the same community could be overlapped. Therefore, EDRL-IM has a better performance than CMA-IM.

To investigate the convergence of EDRL-IM, Fig. 8 shows the variation of $\mathbf{F}$ values with the number of iterations on the Poolbooks, Email and Wiki networks. The results show that the $\mathbf{F}$ values are increased with the increasing number of iterations when $n_g \leq 60$, which validates the exploration ability of EDRL-IM over the evolution. They also present that EDRL-IM can converge to good solutions within 100 iterations, which demonstrates the convergence and exploitation ability of EDRL-IM in solving the IM problem.

VI. Conclusion

In this article, we have modeled the influence maximization of complex networks as the continuous weight parameter optimization of a deep Q network, and have proposed an evolutionary deep reinforcement learning algorithm (called EDRL-IM) for solving the modeled optimization problem. In EDRL-IM, an evolutionary algorithm has been designed to evolve simultaneously a population of DQNs, each of which can make decisions for the selection of seed nodes through a dynamic markov node selection strategy, while a deep reinforcement learning algorithm has been proposed to accelerate the evolution by using an SGD optimization strategy and integrating all DQNs' training information to optimize the DQN. Experiments on both the GN and LFR benchmark networks and ten real-world networks have shown the advantages of EDRL-IM over the state-of-the-art IM algorithms for finding the influential seed node set, and have validated the generalization of EDRL-IM for tackling different types of networks.

This work was done under the IC influence propagation model, which may be unsuitable certain real scenery such as the influence spreading of social systems under the novel coronavirus (COVID-19), Zika virus and other infectious diseases. In this case, as part of our future work, we will study the IM problem in complex networks under various influence propagation models such as LT, TD, SIR, SEIR, SIQR and SEIQR models, and will use our EDRL-IM solution for tasks of infectious diseases such as disease prediction, target immunity, and analysis of epidemic prevention measures. Moreover, we will study the parallelization of the proposed EDRL-IM, aiming to solve the IM problem in the super-large scale networks which have millions of nodes and billions of edges. In addition, we will extend the proposed EDRL-IM for solving the IM problem in multilayer, temporal, and dynamic networks under with heterogeneous and dynamic influence propagation probability. Finally, inspired by the works [52], we will propose a multitasking evolutionary DRL for simultaneously solving multiple similar optimization problems in complex networks.

References

[1] Y. Li, J. Fan, Y. Wang, and K. L. Tan, "Influence maximization on social graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 10, pp. 1852–1872, Oct. 2018.

[2] Z. Zhang, H. Wang, C. Wang, and H. Fang, "Modeling epidemics spreading on social contact networks," *IEEE Trans. Emerg. Top. Comput.*, vol. 3, no. 3, pp. 410–419, Sep. 2015.

[3] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, Washington, DC, USA, ACM, 2010, pp. 1029–1038.

[4] J. Kim, S. K. Kim, and H. Yu, "Scalable and parallelizable processing of influence maximization for large-scale social networks," in *Proc. IEEE 29th Int. Conf. Data Eng.*, Brisbane, Australia, 2013, pp. 266–277.

[5] F. Morone and H. A. Makse, "Influence maximization in complex networks through optimal percolation," *Nature*, vol. 524, no. 7563, pp. 65–68, 2015.

[6] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, Washington, DC, USA, ACM, 2003, pp. 137–146.

[7] M. Gong, C. Song, C. Duan, L. Ma, and B. Shen, "An efficient memetic algorithm for influence maximization in social networks," *IEEE Comput. Intell. Mag.*, vol. 11, no. 3, pp. 22–33, Aug. 2016.

[8] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, San Francisco, CA, USA, ACM, 2001, pp. 57–66.

[9] Z. Du, X. Xu, Y. Wu, L. Wang, B. J. Cowling, and L. A. Meyers, "Serial interval of COVID-19 among publicly reported confirmed cases," *Emerg. Infect. Dis.*, vol. 26, no. 6, pp. 1341–1343, 2020.

[10] D. Duque, D. P. Morton, B. Singh, Z. Du, R. Pasco, and L. A. Meyers, "Timing social distancing to avert unmanageable COVID-19 hospital surges," *Proc. Nat. Acad. Sci.*, vol. 117, no. 33, pp. 19873–19878, 2020.

[11] C.-H. Cheng, Y.-H. Kuo, and Z. Zhou, "Outbreak minimization vs influence maximization: An optimization framework," *BMC Med. Informat. Decis. Mak.*, vol. 20, no. 1, pp. 1–13, 2020.

[12] P. Li, K. Liu, K. Li, J. Liu, and D. Zhou, "Estimating user influence ranking in independent cascade model," *Physica A, Stat. Mech. Appl.*, vol. 565, 2021, Art. no. 125584.

[13] F. Riquelme, P. Gonzalez-Cantergiani, X. Molinero, and M. Serna, "Centrality measure in social networks based on linear threshold model," *Knowl.-Based Syst.*, vol. 140, pp. 92–102, 2018.

[14] W. Chen, W. Lu, and N. Zhang, "Time-critical influence maximization in social networks with time-delayed diffusion process," in *Proc. AAAI Conf. Artif. Intell.*, Toronto, Ontario, Canada, AAAI, 2012, vol. 26, no. 1, pp. 592–598.

[15] A. Goyal, W. Lu, and L. V. Lakshmanan, "Celf optimizing the greedy algorithm for influence maximization in social networks," in *Proc. 20th Int. Conf. Companion World Wide Web*, Hyderabad, India, ACM, 2011, pp. 47–48.

[16] Y. Wang, G. Cong, G. Song, and K. Xie, "Community-based greedy algorithm for mining top-K influential nodes in mobile social networks," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, Washington, DC, USA, ACM, 2010, pp. 1039–1048.

[17] X. Liu, M. Li, S. Li, S. Peng, X. Liao, and X. Lu, "IMGPU: GPU-accelerated influence maximization in large-scale social networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 136–145, Jan. 2014.

[18] X. Yin, X. Hu, Y. Chen, X. Yuan, and B. Li, "Signed-PageRank: An efficient influence maximization framework for signed social networks," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 5, pp. 2208–2222, May 2021.

[19] G. Tong and R. Wang, "On adaptive influence maximization under general feedback models," *IEEE Trans. Emerg. Top. Comput.*, 2020, to be published, doi: 10.1109/TETC.2020.3031057.

[20] T. Cai, J. Li, A. S. Mian, T. Sellis, J. X. Yu, "Target-aware holistic influence maximization in spatial social networks," *IEEE Trans. Knowl. Data Eng.*, 2020, to be published, doi: 10.1109/TKDE.2020.3003047.

[21] C. Feng *et al.*, "Neighborhood matters: Influence maximization in social networks with limited access," *IEEE Trans. Knowl. Data Eng.*, 2020, to be published, doi: 10.1109/TKDE.2020.3015387.

[22] M. Gong, J. Yan, B. Shen, L. Ma, and Q. Cai, "Influence maximization in social networks based on discrete particle swarm optimization," *Inf. Sci.*, vol. 367, pp. 600–614, 2020.

[23] A. ŞİMŞEK and K. Resul, "Using swarm intelligence algorithms to detect influential individuals for influence maximization in social networks," *Expert Syst. Appl.*, vol. 114, pp. 224–236, 2018.

[24] L. Cui *et al.*, "DDSE: A novel evolutionary algorithm based on degree-descending search strategy for influence maximization in social networks," *J. Netw. Comput. Appl.*, vol. 103, pp. 119–130, 2018.

[25] S. Wang, J. Liu, and Y. Jin, "Finding influential nodes in multiplex networks using a memetic algorithm," *IEEE Trans. Cybern.*, vol. 51, no. 2, pp. 900–912, Feb. 2021.

[26] L. Zhang, L. Yutong, F. Cheng, J. Qiu, and X. Zhang, "A local-global influence indicator based constrained evolutionary algorithm for budgeted influence maximization in social networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1557–1570, Apr.–Jun. 2021.

[27] Y. Hou, Y.-S. Ong, L. Feng, and J. M. Zurada, "An evolutionary transfer reinforcement learning framework for multiagent systems," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 601–615, Aug. 2017.

[28] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Curran Associates, Inc., 2017, vol. 30, pp. 6348–6358.

[29] S. Manchanda *et al.*, "GCOMB: Learning budget-constrained combinatorial algorithms over billion-sized graphs," *Proc. Adv. Neural Inf. Process. Syst.*, Curran Associates, Inc., vol. 33, pp. 20000–20011, 2020.

[30] H. Li, M. Xu, S. S. Bhowmick, C. Sun, Z. Jiang, and J. Cui, "DISCO: Influence maximization meets network embedding and deep learning," 2019, *arXiv:1906.07378*.

[31] C. Fan, L. Zeng, Y. Sun, and Y.-Y. Liu, "Finding key players in complex networks through deep reinforcement learning," *Nat. Mach. Intell.*, vol. 2, no. 6, pp. 317–324, 2020.

[32] J.-R. Lee and C.-W. Chung, "A fast approximation for influence maximization in large social networks," in *Proc. 23rd Int. Conf. World Wide Web*, Seoul, Republic of Korea, ACM, 2014, pp. 1157–1162.

[33] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, New York, NY, USA, ACM, 2014, pp. 701–710.

[34] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, San Francisco, CA, USA, ACM, 2016, pp. 855–864.

[35] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, San Francisco, CA, USA, ACM, 2016, pp. 1225–1234.

[36] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, Florence, Italy, ACM, 2015, pp. 1067–1077.

[37] F. Niu, B. Recht, C. Ré, and S. J. Wright, "HOGWILD!: A lock-free approach to parallelizing stochastic gradient descent," *Proc. Adv. Neural Inf. Process. Syst.*, Granada, Spain, Curran Associates, Inc., vol. 24, pp. 693–701, 2011.

[38] S. Khadka and K. Tumer, "Evolution-guided policy gradient in reinforcement learning," *Proc. Adv. Neural Inf. Process. Syst.*, Montréal, Canada, Curran Associates, Inc., vol. 31, pp. 1188–1200, 2018.

[39] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proc. Nat. Acad. Sci.*, vol. 99, no. 12, pp. 7821–7826, 2002.

[40] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Phys. Rev. E*, vol. 80, no. 1, 2009, Art. no. 016118.

[41] L. Ma, J. Li, Q. Lin, M. Gong, C. A. C. Coello, and Z. Ming, "Reliable link inference for network data with community structures," *IEEE Trans. Cybern.*, vol. 49, no. 9, pp. 3347–3361, Sep. 2019.

[42] M. Gong, Q. Cai, X. Chen, and L. Ma, "Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 82–97, Feb. 2014.

[43] P. M. Gleiser and L. Danon, "Community structure in jazz," *Adv. Complex Syst.*, vol. 6, no. 04, pp. 565–573, 2003.

[44] G. Guo, J. Zhang, and N. Yorke-Smith, "A novel Bayesian similarity measure for recommender systems," *Proc. 23rd Int. Joint Conf. Artif. Intell.*, Beijing, China, IJCAI/AAAI, vol. 13, pp. 2619–2625, 2013.

[45] R. Guimera, L. Danon, A. Diaz-Guilera, F. Giralt, and A. Arenas, "Self-similar community structure in a network of human interactions," *Phys. Rev. E*, vol. 68, no. 6, 2003, Art. no. 065103.

[46] J. Kunegis, "KONECT: the Koblenz network collection," in *Proc. 22nd Int. Conf. World Wide Web*, Rio de Janeiro, Brazil, 2013, pp. 1343–1350.

[47] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[48] L. Ma, Y. Ma, Q. Lin, J. Ji, C. A. C. Coello, and M. Gong, "SNE-GAN: Signed network embedding by using generative adversarial nets," *IEEE Trans. Emerg. Top. Comput. Intell.*, 2020, to be published, doi: 10.1109/TETCI.2020.3035937.

[49] M. Boguná, R. Pastor-Satorras, A. Díaz-Guilera, and A. Arenas, "Models of social networks based on social distance attachment," *Phys. Rev. E*, vol. 70, no. 5, 2004, Art. no. 056122.

[50] L. Ma, M. Gong, J. Liu, Q. Cai, and L. Jiao, "Multi-level learning based memetic algorithm for community detection," *Appl. Soft Comput.*, vol. 19, no. 2, pp. 121–133, 2014.

[51] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, San Jose, California, USA, ACM, 2007, pp. 420–429.

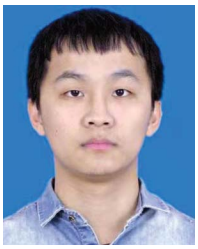
[52] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution: Toward evolutionary multitasking," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 343–357, Jun. 2016.

Lijia Ma received the B.S. degree in communication engineering from Hunan Normal University, Changsha, China, and the Ph.D. degree in electronic science and technology from Xidian University, Xi'an, China, in 2010 and 2015, respectively.

From 2015 to 2016, he was a Postdoctoral Fellow with Hong Kong Baptist University, Hong Kong, and from 2016 to 2017, he was with Nanyang Technological University, Singapore. He is currently an Assistant Professor with the College of Computer and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include evolutionary computation, machine learning, and complex networks.

Zengyang Shao received the B.S. degree from Shenzhen University, Shenzhen, China, in 2019. He is currently working toward the master's degree with the College of Computer and Software Engineering.

His research interests include evolutionary computation, machine learning, and complex networks.



Xiaocong Li received the B.S. degree from Shenzhen University, Shenzhen, China, in 2020. He is currently working toward the master's degree with the College of Computer and Software Engineering.

His research interests include evolutionary computation, machine learning, and complex networks.

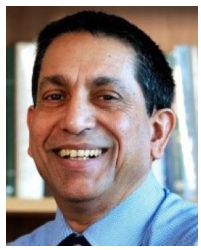
Qiuzhen Lin (Member, IEEE) received the B.S. degree from Zhaoqing University, Zhaoqing, China, in 2007, the M.S. degree from Shenzhen University, Shenzhen, China, in 2010, and the Ph.D. degree from the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, in 2014, respectively.

He is currently an Associate Professor with the College of Computer Science and Software Engineering, Shenzhen University. His current research interests include artificial immune system, multi-objective optimization, and dynamic system.

Jianqiang Li (Member, IEEE) received the B.S. and Ph.D. degrees in automation major from the South China University of Technology, Guangzhou, China, in 2003 and 2008, respectively.

He is currently a Professor with the College of Computer and Software Engineering, Shenzhen University, Shenzhen, China. He led three projects with the National Natural Science Foundation of China. His research interests include embedded systems and Internet of Things.

Victor C. M. Leung (Fellow, IEEE) received the B.A.Sc. (Hons.) degree in electrical engineering from the University of British Columbia (UBC), Vancouver, BC, Canada, in 1977. He attended graduate school with UBC on a Canadian Natural Sciences and Engineering Research Council Postgraduate Scholarship and received the Ph.D. degree in electrical engineering in 1982. From 1981 to 1987, he was a Senior Member with Technical Staff and Satellite System Specialist, MPR Teltech Ltd., Canada. He started his academic career with the Department of Electronics, Chinese University of Hong Kong, Hong Kong, in 1988. He returned to UBC as a Faculty Member in 1989, where he currently holds the positions of a Professor and a TELUS Mobility Research Chair in Advanced Telecommunications Engineering with the Department of Electrical and Computer Engineering. He is also the Distinguished Professor with the School of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. He has coauthored more than 1000 journal/conference papers, 37 book chapters, and coedited 12 book titles. Dr. Leung is a registered Professional Engineer in the Province of British Columbia, Canada. He was awarded the APEBC Gold Medal as the Head of the graduating class in the Faculty of Applied Science



Asoke K. Nandi (Fellow, IEEE) received the Ph.D. degree in physics from the University of Cambridge (Trinity College), Cambridge U.K. He held academic positions in several universities, including the University of Oxford, Oxford, U.K., Imperial College London, London, U.K., University of Strathclyde, Glasgow, U.K., and University of Liverpool, Liverpool, U.K., and also with Finland Distinguished Professorship in Jyvaskyla, Finland. In 2013, he moved to Brunel University London, London, U.K., to become the Chair and the Head of electronic and computer engineering. He is currently a Distinguished Visiting Professor with Shenzhen University, Shenzhen, China, and an Adjunct Professor with the University of Calgary, Calgary, AB, Canada.

He has authored more than 600 technical publications, including 250 journal papers and five books, entitled *Condition Monitoring with Vibration Signals: Compressive Sampling and Learning Algorithms for Rotating Machines* (Wiley, 2020), *Automatic Modulation Classification: Principles, Algorithms and Applications* (Wiley, 2015), *Integrative Cluster Analysis in Bioinformatics* (Wiley, 2015), *Blind Estimation Using Higher-Order Statistics* (Springer, 1999), and *Automatic Modulation Recognition of Communications Signals* (Springer, 1996). The H-index of his publications is 80 (Google Scholar) and his ERDOS number is two. His research interests include signal processing and machine learning, with applications to communications, image segmentations, and biomedical data. He has made many fundamental theoretical and algorithmic contributions to many aspects of signal processing and machine learning. He has much expertise in Big and Heterogeneous Data, dealing with modeling, classification, estimation, and prediction.

In 1983, he co-discovered the three fundamental particles known as W^+, W^- and Z^0 (by the UA1 team at CERN), providing the evidence for the unification of the electromagnetic and weak forces, for which the Nobel Committee for Physics in 1984 awarded the prize to his two team leaders for their decisive contributions. Professor Nandi is a Fellow of the Royal Academy of Engineering (U.K.) and also a Fellow of seven other institutions, including the IET. Among the many awards, he was the recipient of the Institute of Electrical and Electronics Engineers (USA) Heinrich Hertz Award in 2012, the Glory of Bengal Award for his outstanding achievements in scientific research in 2010, the Water Arbitration Prize of the Institution of Mechanical Engineers (U.K.) in 1999, and the Mountbatten Premium, Division Award of the Electronics and Communications Division, Institution of Electrical Engineers (U.K.) in 1998. Professor Nandi was an IEEE EMBS Distinguished Lecturer during 2018–2019.