Received: January 24, 2024. Revised: April 20, 2024. 

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

# Adaptive Hybridization of Differential Evolution and DQN - Reinforcement Learning to Solve the Influence Maximization Problem in Social Networks

**Agash Uthayasuriyan[1] Hema Chandran G[2] Kavvin UV[2] Sabbineni Hema Mahitha[2] Jeyakumar G[3]***

*[1]Department of Multidisciplinary Graduate Engineering, College of Engineering, Northeastern University, Boston, USA*
*[2]Department of Electrical and Electronics Engineering, Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, India*
*[3]Department of Computer Science and Engineering, Amrita School of Computing, Coimbatore, Amrita Vishwa Vidyapeetham, India*
* Corresponding author's Email: g_jeyakumar@cb.amrita.edu

**Abstract:** Understanding the dynamics of information dissemination is essential in the rapidly changing world of social networks. The Influence Maximization (IM) problem, which seeks to identify key nodes in a network for optimal information spread, has been a focal point of research. This paper introduces the DERL (Differential Evolution - Reinforcement Learning) framework, a novel approach that seamlessly integrates Differential Evolution (DE) and DQN-based Reinforcement Learning (DQN-RL) to address the IM problem. By harnessing the optimization capabilities of DE and the adaptive learning strengths of DQN-RL, the DERL framework presents an adaptive hybrid solution to the IM problem. The proposed framework is tested alongside established models using benchmark datasets. The results denote that the DERL framework excels over compared models, achieving 2.5 times higher fitness scores in Influence Maximization. The study underscores the potential of interdisciplinary approaches in deciphering complex network challenges and also solves the IM problem effectively.

**Keywords:** Social network, Influence maximization, Line embedding, Differential evolution, DQN - reinforcement learning and evolutionary learning.

## 1. Introduction

Large-scale social and complex networks have emerged as a result of technological advancements, comprising of thousands of interconnected nodes. Analysing these network structures is crucial [1] in identifying the communities [2], future possible connections between nodes [3] and also in identifying the network's most influential nodes [4].

IM is one of the most researched areas in social network [5,6], which aims to find a subset of nodes in a network (seed nodes) that, when activated or influenced, would result in the maximum spread of information or influence throughout the network. Solving the IM problem has practical implications in fields such as marketing, viral advertising, opinion formation, and even in public health campaigns. By understanding the dynamics of influence within social networks and identifying key individuals who can act as influential spreaders, it becomes possible to design more effective strategies for information dissemination and behaviour change in a targeted manner.

The IM problem is considered to be NP-hard, and therefore finding the optimal solution becomes computationally challenging and time-consuming as the network size grows. Various mathematical, Evolutionary Algorithms (EA) [7] based and Machine Learning (ML) based models have been developed by researchers to tackle this problem.

EAs, the most preferred models to solve the optimization problems are inspired from the

Darwin's ideology of "Survival of the fittest", where the algorithm begins from a random set of population (solution set) and improves them continuously across iterations in arriving at a better solution. Among the present EA models, Differential Evolution (DE) is one of the most powerful algorithms that uses a unique mutation technique and has proved itself in solving many optimization problems. On the other hand - ML, a branch of Artificial Intelligence (AI) learns and analyses the patterns present in the data provided to produce results. Reinforcement Learning (RL), a part of ML paradigm involves training agents to make sequential decisions in an environment, with feedback provided through rewards or penalties. This paper aims at leveraging the strengths of EA and ML, in solving the IM problem and the main contribution of the study can be summarised as follows.

1. Hybrid Algorithm (DERL): The paper introduces a novel hybrid framework called DERL (Differential Evolution and DQN-based Reinforcement Learning), that combines techniques from the Evolutionary Algorithm domain (Differential Evolution) and the Machine Learning domain (DQN-based Reinforcement Learning) to identify influential seed nodes in social network. This hybrid framework integrates two different paradigms to address the influence maximization (IM) problem.

2. Adaptive Fitness Transfer: The DERL framework incorporates an adaptive fitness transfer technique. This technique improves the solution set of one algorithm based on the best individual solution obtained by the other algorithm. This adaptive transfer mechanism enhances the overall performance of the hybrid algorithm, making it more effective in identifying influential nodes within the network.

3. Validation & Evaluation of DERL framework: The paper evaluates the DERL framework by comparing its performance to standalone DE and DQN-RL models on a selection of seven carefully chosen network datasets. Additionally, the hybrid framework is also compared with other established models using benchmark networks and the results demonstrate that the adaptive hybridization technique of DERL outperforms both standalone DE and DQN-RL models, as well as other existing models, showcasing the efficacy of combining diverse computational paradigms.

These contributions are expected to collectively advance the field of influence maximization in social networks and provide valuable insights for researchers and practitioners in this area. The remaining part of the paper is organized as follows. Section 2 presents the work related to solving IM by using EA and RL. Section 3 explains the proposed framework carried out in this study, followed by Section 4, that provides the experimental design. The results recorded and the inferences obtained are presented in Section 5, and finally, Section 6 concludes the paper.

## 2. Related works

The recent studies along with few other notable researches that discuss solving the IM problem using mathematical models, optimization algorithms, reinforcement learning techniques, and hybrid algorithms developed by various authors have been summarized in this section.

In [8], the authors have presented an adaptive probability-based evolutionary method based on the topic affinity propagation (TAP) method, that determines the ideal collection of important nodes in the network. The TAP technique's efficiency is evaluated on two networks of large scale. In comparison to different IM strategies, results show that the suggested algorithm helps to enhance the influence propagation.

The article presented in [9] delves into maximizing influence spread through social networks, examining various diffusion models like the Linear Threshold and Independent Cascade models. These models explore how innovations or behaviours spread across a network, focusing on the role of strategically influential nodes. The authors use submodular function theory to offer the first approximation guarantees for influence maximization, proving that a simple greedy algorithm can achieve close to optimal solutions.

A mathematical framework is proposed by authors of [10] to solve the problem of identifying the minimal set of influential nodes in random networks, essential for optimizing information spread or immunization strategies. This involves mapping the problem onto optimal percolation, using the non-backtracking matrix of the network to minimize the energy of the system. The results reveal that this method identifies previously overlooked low-degree nodes as crucial influencers, contrary to traditional methods that focus on highly connected nodes. These low-degree nodes significantly enhance network efficiency in spreading processes or immunization against epidemics.

CELF (Cost Effective with Lazy Forward), presented by [11] extends the concept of node degrees by incorporating the influence spread of nodes. It starts by calculating the influence spread range for each node and selects the node with the highest influence spread as a seed node. It then updates the influence spread of nodes in

submodularities containing seed nodes iteratively until all seed nodes are chosen.

The work in paper [12] proposes CDDE (discrete differential evolution algorithm based on community structure). The Louvain approach initially utilized to determine the community structure and notable communities are selected on this basis, and suitable nodes are extracted out of every community. According to experimental results on six real-world social networks, the CDDE model produces equivalent influence dispersion to CELF and competes with comparable methods in terms of efficacy and cost effectiveness.

Research work in [13] proposes DPSO (discrete particle swarm optimization) approach, that makes search in the selection space more efficient. Experiments on four real-world social networks show that the proposed algorithm for impact maximization is both practical and effective. In paper [14], the authors discuss DDSE (degree-descending search evolution), that prevents the performance difficulty of greedy methods through removing recurred exercises using EDV (estimation function). Results from tests on actual social networks demonstrate that DDSE is about five orders of magnitude faster than the most advanced greedy method while maintaining competitive accuracy, confirming the method's enormous utility and effectiveness.

The research in [15] introduces a memetic algorithm, named CMA-IM, for influence maximization in social networks. The methodology leverages a two-step process that includes network clustering to identify communities and a candidate selection phase to pinpoint influential nodes. This algorithm specifically optimizes a 2-hop influence spread, integrating a genetic algorithm for global search and similarity-based strategies for local search. Results show superior performance compared to classical methods like Degree, in achieving higher influence spread more efficiently.

MA-IMmulti in [16] is an extension of CMA-IM designed to handle multiplex networks. It approaches the IM problem by optimizing multiplex 2-hop influence spread and introduces a novel memetic algorithm to solve it. The initial seed sets are generated using a combination of random, roulette-based degree, and distance selection strategies.

The research presented in [17] demonstrates that when the Influence Maximization problem is solved using evolutionary algorithms such as Genetic Algorithm, Ant Colony Optimization, Differential Evolution and Particle Swarm Optimization, Differential Evolution outperformed others in terms of solution quality and algorithmic efficiency.

Paper [18] introduces an evolutionary Transfer RL framework (eTL) for MASs, drawing from Darwin's natural selection and Universal Darwinism. This framework, featuring memetic automaton and mechanisms like meme assimilation and evolution, addresses current TL (Transfer Reinforcement Learning) limitations such as reliance on poor advice. Empirical validation through minefield navigation and "Unreal Tournament 2004" scenarios demonstrates its superiority over existing TL methods, proving effective in developing adaptive agents for complex environments.

FINDER [19], a deep reinforcement learning framework designed for identifying key players in complex networks to enhance or degrade network functionality. It models node selection as a Markov decision process, employing graph representation learning to encode network structures and reinforcement learning to determine optimal node removal sequences.

The authors of S2V-DQN [20] present a method of that employs the structure2vec method to embed networks into low-dimensional node representations. It then uses Deep Reinforcement Learning (DRL) with ε-degree samples to train a Deep Q Network (DQN) and generates seed sets based on the seed scores produced by the DQN.

The article [21] presents a novel approach to influence maximization in complex networks by employing an evolutionary deep reinforcement learning algorithm (EDRL-IM). The proposed methodology integrates evolutionary algorithms (EA) with deep reinforcement learning (DRL) to optimize a deep Q network for selecting influential seed nodes in a network. EDRL-IM starts with the creation of various potential solutions modelled as deep Q networks, which are then evolved using both EA and DRL strategies, enhancing the efficiency of traditional influence maximization methods.

The authors in paper [22] presents a PIANO model which combines embedding of the network and Reinforcement Learning to deal with IM problem, and they additionally proposed PIANO-E and PIANO @d, the two of which are able to be utilized precisely to answer IM regardless of training the algorithm. The experiment conducted on networks indicates that the model outperforms state-of-the-art classical approaches in terms of efficiency and impact distribution accuracy.

The study in paper [23] addresses influence maximization (IM) in online social networks, a problem initially formulated as a combinatorial optimization challenge. Traditional approaches, including Greedy algorithms and reverse influence sampling, faced limitations in accurately estimating

influence spread. The paper introduces ToupleGDD, a novel deep reinforcement learning framework combining Graph Neural Networks (GNNs) and Double Deep Q-networks (DDQN). This framework, distinct for its personalized node embedding and focus on diffusion cascades, demonstrates enhanced performance in accurately solving the IM problem, outperforming existing algorithms in tests on various datasets.

The paper [24] introduces a novel approach called Multi-Transformation Evolutionary Framework for IM (MTEFIM). This framework utilizes multiple proxy models to improve the performance of evolutionary-based IM algorithms. It incorporates a knowledge transfer process across transformations to leverage common information adaptively. MTEFIM provides a comprehensive solution by considering the optimal seed set from each transformation.

After analysing the performance of existing mathematical models, optimization algorithms and reinforcement learning techniques for solving the learning and optimization problems around us, this paper attempts to propose a framework to hybridise a learning algorithm and an optimization algorithm. For the design of the proposed framework, the existing hybrid algorithms were also reviewed. The detailed description about the proposed framework is presented in Section 3.

The intention of the proposed research problem is to investigate the performance difference of the hybrid framework comparing to the constituent EA and ML algorithms.

## 3. Proposed framework

Identifying a particular set of people or nodes within a network, known as "seed nodes" is necessary to solve the IM problem. When influencing others in the network, these seed nodes are carefully chosen to maximize the overall impact of advertising campaigns. The goal is to choose the seed nodes in such a way that the network's largest possible population can be influenced.

To identify these seed nodes, a hybrid of two algorithms - Differential Evolution from the EA domain and the DQN - based RL from the ML domain an Evolutionary Learning framework, named as DERL, is proposed in this paper. This DERL framework is designed to use adaptive fitness transfer technique, that improves one algorithm's solution set based on the other algorithm's solution.

To identify influential nodes in a network, a fitness function is required to evaluate the scores of the generated seed set. There are various influence propagation models available, such as the Linear Threshold (LT) [25], Independent Cascade (IC) [26], and other diffusion models [27]. The proposed study focuses on using the IC model, but the approach can be extended to other models as well.

The "word-of-mouth" phenomenon, in which people are influenced by the opinions, decisions, and actions of their friends and neighbours, is captured by the IC model. Each node in the network has the potential to activate its neighbours in accordance with the IC model, based on the edge propagation probability. This activation process is repeated indefinitely until no more nodes are activated. Performing standard IC propagation steps on a large-scale network can take time, especially during the final propagation phase, which may involve a large number of activations.

To address this, a fast approximation IC model was proposed by Lee and Chung [28]. This model simplifies the propagation process by limiting the influence ability of seed nodes to other nodes that are in 2-hop range (neighbours' neighbours). In this approximation, the final propagation size is no greater than 2. The influence spread of a specific seed set can be accurately measured by this fast approximation IC model, also known as FastIM, according to extensive analysis and tests.

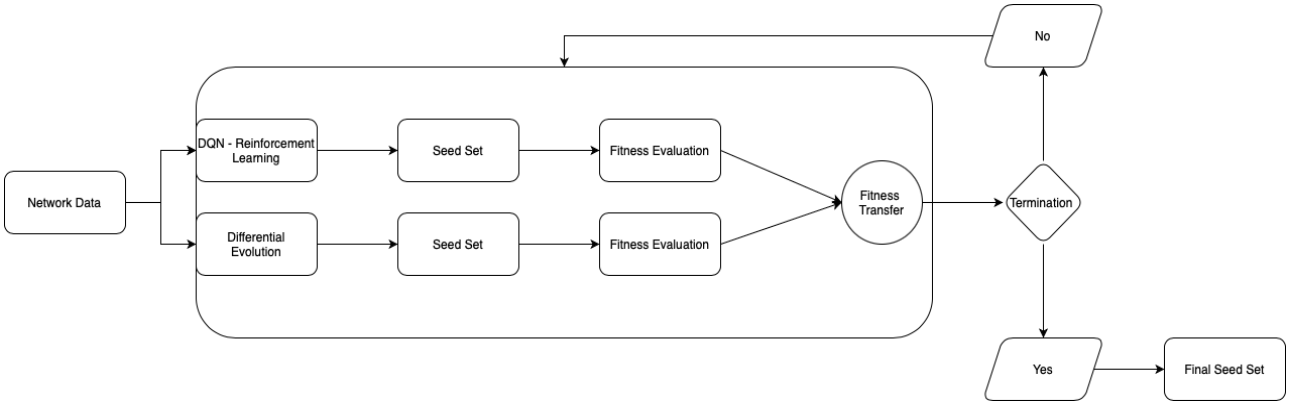


Figure. 1 Workflow of DERL Hybrid framework

FastIM is therefore selected as the fitness function in this study, since it provides quicker execution and reliable results.

In adaptive hybridization, experimented in the proposed DERL framework, the solution isn't relied on a single algorithm. The DE and DQN - RL can be run for a certain number of iterations in each epoch. After the end of each epoch, the fitness obtained by both the algorithms can be combined, ranked and the best obtained fitness value along with the seed nodes can be inserted into the population of the algorithms, that makes the algorithms continue their search of solutions from a more informed or upgraded solution.

By employing this method, we can achieve adaptive hybridization where the weaker epoch of one algorithm (based on obtained fitness value) is compensated by the other algorithm and, is also improved in the next epoch based on the other algorithm's obtained seed set. A high-level workflow diagram of the proposed framework is presented in Fig. 1.

Brief descriptions of the standalone DE, DQN-RL algorithm, and DERL hybrid framework's design are provided in the following subsections, which address the IM problem, having the "FastIM" as the fitness function to evaluate the produced solutions.

### 3.1 Differential evolution

Evolutionary Algorithms, a part of Evolutionary Computation paradigm is inspired by the principle of natural evolution. These EAs operate by iteratively applying the genetic operations such as selection, mutation and re-combination. The key advantage of EAs is their ability to search through large and complex solution set and by not limiting itself any local optimal solution. For a given objective function, the EAs populates its solution set to attain the maximum/ minimum value.

The DE algorithm is considered as a powerful EA, mainly because of the distinctive mutation operation, that allows for wide exploration of the solution space. This operator combines candidate solutions with the existing population, promoting diverse solutions and enabling DE to escape local optima. The mutation process in DE also ensures a good balance between exploration and exploitation, allowing it to converge towards optimal solutions while maintaining population diversity. Furthermore, DE has shown superior performance in terms of convergence speed and solution quality compared to other EAs in many benchmark problems.

In the view of solving the IM problem, the DE is designed to start with a random set of initial seed nodes based on the pre-defined population size. This population set keeps getting refined over time across iterations based on the fitness function through mutation, crossover and selection techniques. The mutation step as shown in Eq. (1), introduces diversity into the individuals, making the algorithm explore search space.

$$Mutated\ vector = X1 + F\,(X2 - X3) \quad (1)$$

In this experiment, the standard DE/rand/1 mutation operation has been followed, that selects any three solutions from the population and calculates the new vector by scaling the difference of two vectors. The presence of scaling factor "F" ensures the amount of difference to be added. A larger value of "F" promotes increased exploration of the solution space, allowing for a broader search and consideration of diverse solutions, while a smaller value of "F" encourages greater exploitation.

Followed by mutation, the population is made to crossover that enables the merging of two solutions, to generate a new one. The crossover rate, denoted as "CR", determines the likelihood of selecting a portion of the mutated solution, enabling exploration within the solution space. Subsequently, the resulting solution is included in the new population, as shown in Eq. (2).

$$\text{Vector} = \begin{cases} \text{Mutated Vector if Random Probability} \leq \text{CR} \\ \text{Existing Vector if Random Probability} > \text{CR} \end{cases} \quad (2)$$

After the entire population undergoes mutation and crossover operations, the selection operation is done, where each solution is ranked based on its fitness value calculated using the "FastIM" fitness function. This step ensure that the best solutions are preserved and improved upon the next generation. The global best position and fitness are updated if the maximum personal best fitness score is higher than the global best fitness score.

To maintain presence of the best solutions and prevent the algorithm from exploring inferior solutions, the top solutions based on a chosen elitism value are preserved and transferred to the population in the next iteration. This ensures that the best solutions persist throughout the optimization process. This iterative process continues until the termination criteria, such as a specific number of iterations or reaching a desired fitness threshold, are met. Finally, the best seed set and its corresponding influence score are returned.

### 3.2 DQN - reinforcement learning

Reinforcement Learning (RL) - an unsupervised machine learning algorithm, enables the system to understand the interaction between agent and environment to maximize the reward of the agent. The agent works based on the reward offered by the user for its successful action. Reinforcement learning can be used to find influential seed nodes that maximize the network's ability to disseminate information in order to solve the problems that arise in social networks [29] and IM [30]. Reinforcement learning are highly adaptive and dynamic in nature. The agent can learn optimal strategies for selecting nodes based on outcomes of previous selections.

Deep Q Network RL combines deep neural networks and Q learning to achieve optimal state-action value function based on the Q value. It is one of the popular methods for solving complex sequential decision-making problems [31]. DQN RL can handle complex network structures and large-scale data sets. They can effectively explore the right seed nodes in the social network and maximize the spread of influence. They can also handle sequential decision-making tasks and can optimize the selection process. The general terminologies involved in the DQN-RL algorithm and a short note on them are as follows.

**Agent:** The reinforcement learning agent refers to the entity that interacts with the environment and learns to make decisions based on various parameters. In the context of IM, the agent's goal is to select the set of influential nodes in a network to maximize the spread of information.

**Environment:** The place or domain where the agent works is referred as the environment. In the context of IM, the network data is the environment containing nodes and edges. It provides information such as current state (current seed node), and enables agent to take action and provides rewards.

**State:** The state is the information that represents the current position or configuration of the environment.

**Action:** Action are the decisions that the agent can take in the environment. Relating with IM, an action corresponds to selecting a node from the available nodes in the network to activate and aim to maximize the spread of influence.

**Reward:** The reward is a value that provides feedback to the agent based on its actions. In IM, the reward is given to the agent to measure the quality of the selected set. It may be based on number of influenced nodes or the influence spread value.

In DQN RL, the agent interacts with an environment and learns to take actions in order to maximize reward. The agent observes the current node in the network, selects a seed node and receives a reward along with the next state. The main goal of the agent is to learn a policy that maximizes the expected reward over time. Applying Deep Q Network handles high dimensional state spaces with deep neural networks. But Q values cannot be stored as a tabular representation since the size of real-life network will be really high. Instead, Q values are stored in the form of state action pair, and DQN gets the Q values using neural networks which can handle real life complex data sets and continuous state spaces. The Q value function is learned by updating the network parameters iteratively using the Bellman equation, as represented in Eq. (3).

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha.\,[R_{t+1} + \gamma.\,max_a\, Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (3)$$

The parameters 's' represents state i.e., current seed node, 'a' represents action selecting next node, "R" for the unbiased true reward. The "Learning rate (α)" decides the step size at which the neural network updates its parameter during training. The relative importance of long-term rewards versus short-term rewards is determined by the "Discount Factor (γ)". It has a value between 0 and 1, with 0 indicating that the agent only takes into account immediate rewards and 1 indicating that the agent values all future rewards equally. Apart from these parameters involved in the Bellman's equation, DQN RL involves a few hyperparameters which need to be set properly for effective learning of the neural network.

The number of experiences sampled from the replay memory during training for each iteration is represented by the parameter "Batch Size". The batch size should be carefully chosen because although a larger batch size can result in more stable updates, it also necessitates more computation. For training purposes, the agent's previous actions and experiences are stored in a buffer called "Replay Buffer Size". A larger replay buffer size can result in better learning, but it also requires more memory. It determines the number of experiences stored.

### 3.3 Adaptive hybrid framework - DERL

Evolutionary Learning paradigm involves using a hybrid of Machine Learning and Evolutionary Algorithm to achieve a higher accuracy / fitness in solving a particular problem. This approach aims to combine the strengths of both fields to achieve higher accuracy or fitness in tackling complex problems.

The strength of DQN-RL is that it can learn complex policies through a trial-and-error process,

enabling it to adapt to dynamic and uncertain environments, making it suitable for solving many other problems [32]. The weakness however is the computational expense involved. On the other hand, DE is a population-based optimization algorithm that can quickly converge to optimal or near-optimal solutions, making it suitable for problems with complex search spaces.

1. Function ENV ():
    1. Set nodeNum = number of nodes in the network, seed set size = Number of seed nodes.
    2. Set localInfluenceList = np.zeros(nodeNum)-1,
       OneHopInfluenceList = np.zeros(nodeNum)-1
    3. Construct the graph network and get the network embedded data of the graph network
2. Function AGENT ():
    1. Set population_size=100, buffer_size=10000, learning_times=10.
    2. Initialize population = []
    3. Initialize Reinforcement learning agent as a DQN.
    4. Set gamma=0.8 and learning rate=0.01
    5. Set optimizer as Adam and loss function as MSE.
3. Function EVAL_ALL ():
    1. Initialize all_influence = [], all_seeds = []
    2. For each element in population:
       2.1 Influence, seeds = EVALUATE ()
       2.2 Append fitness to all_influence and seeds to all_seeds
    3. Get best fitness and seeds from all_influence and all_seeds as initial fitness and seeds.
4. Function EVALUATE ():
    1. Initialize total_reward = 0, influence=0, seeds = []
    2. While (len(seeds) < seed set size):
       2.1 Calculate and store Q value of current state by DQN.
       2.2 Sort the Q values in descending order and select the action with highest Q value
       2.3 Append the node corresponding to the action to the seedset.
       2.4 Set next_state = action; calculate reward using 2hop influence method.
       2.5 Update totat_reward = total_reward + reward
       2.6 Store (Q_value, state, next_state, reward) to replay memory.
    3. Return total reward and seeds
5. Function TRAIN ():
    1. Sample random entries of size batch_size from the replay memory. Each entry is of the form {Q_value, State, Next state, Reward}
    2. For each episode of leaning_times:
        For (each entry in batch):
            1. Predict the predicted_state from the Reinforcement Learning DQN forward pass.
            2. Calculate target = Reward + gamma*predicted_state
            3. Calculate target_f by forward pass of DQN from current state.
            4. Calculate loss between target and target_f.
            5. Backpropagate the loss through the DQN network
            6. Evaluate the RL agent.
            7. Return RL agent

(a)

6. Function Mutation (Population, seed set size, F):

    1. Initialize new population = []
    2. While length (new population)! = length (population):
        1. a, b, c = random solutions in population
        2. for i in range (seed set size):
            1. v[i] = a[i] + F * (b[i] – c[i])
            2. v[i] = round(v[i])
        3. new population. append (v)
    3. Return new population

7. Function Crossover (Population, Mutated population, CR):

    1. Initialize new population = []
    2. While length (new population)! = length (population):
        1. i = 0
        2. Probability = random number in range [0,1]
        3. If Probability < = CR:
            1. u = Mutated population [i]
            2. Else: u = Population [i]
        4. new population. append(u)
        5. i+=1
    3. Return new population

8. Function differential_evolution (current_pop, F, CR, max_iter):

    1. For i in range of max_iter:
        1. Population = current_pop
        2. Mutated population = Mutation (Population, seed set size, F)
        3. Crossed population = Crossover (Population, Mutated population, CR)
        4. Rank crossed population based on Fitness function
        5. Replace bottom elitism value of solutions with Elite
        6. If fitness of ($1^{st}$ solution in ranking) > fitness of (Best solution):
            1. Best solution = $1^{st}$ solution in ranking
        7. Update Elite = top elitism value number of solutions
        8. Population = Rank new population based on Fitness function
    2. Return Population [0], Fitness (Population [0]), Population

(b)

9. Initialize environment variables ENV ()
10. Initialize RL agent AGENT ()
11. Initialize Graph
12. Set Fitness = 2hop (seed nodes)
13. current_pop = random DE_Population for given number of population size
14. Set Elite = [] (# top elitism value number of solutions as per given input)
15. Set F = Mutation rate; CR = Crossover rate
16. Evaluate total population to find initial seed set EVAL_ALL ()
17. Set solution_list = [], maxlist = [], next_pop = []
18. global_best_seeds= [], global_best_fitness = 0
19. For (each epoch in total_epoch):
    1. Train RL agent TRAIN ()
    2. Get RL_fitness, average_fitness, RL_seed from training the RL agent.
    3. Append all seeds to solution_list
    4. Append RL_fitness and RL_seed to max_list
    5. DE_seeds, DE_fitness, next_pop = differential_evolution (current_pop, F, CR, 10)
    6. current_pop = next_pop
    7. If DE_fitness>RL_fitness:
        1. Agent (). Population. append (DE_seeds)
        2. If DE_fitness > global_best_fitness:
            1. global_best_fitness = DE_fitness
            2. global_best_seeds = DE_seeds

    else:
        1. Current_pop. append (RL_seeds)
        2. If RL_fitness > global_best_fitness:
            1. global_best_fitness = RL_fitness
            2. global_best_seeds = RL_seeds
20. Return global_best_fitness, global_best_seeds

(c)

Figure. 2 Working of DERL for the functions: (a) ENV, AGENT, EVAL_ALL, EVALUATE and TRAIN, (b) Mn, crossover and differential_evoluation (steps 1 and 2), and (c) Differential_evolution (steps 9 to 20)

It is also observed to be robust to noisy or stochastic environments, making it suitable for real-world applications where data may be corrupted or incomplete. The working methodology of the adaptive hybrid framework - DERL is presented in Fig. 2.

Overall, the strengths and weaknesses of DQN-based RL and DE suggest that a hybridization

approach can leverage the strengths of both algorithms while compensating for their weaknesses [33]. Although, a framework of DE supporting RL in tuning the hyper-parameters or a framework where RL supports DE in learning to select the right values involved in genetic operations can be made and considered a hybrid framework. However, this research aims to use DE and DQN-RL in working together to solve the IM problem by using the adaptive hybridization technique.

The DERL framework runs for 10 epochs and returns the seed set as the final result. Initially, the DERL algorithm starts to work with DQN-RL algorithm in producing the seed set and the RL algorithm is run for 10 iterations for each of DERL framework's epoch. In the working of DQN - RL, function "ENV" initializes the environment variables and constructs the graph network from the dataset. It also initializes the local influence list and one hop influence list that is used to calculate the influence of the nodes of the network. Function "AGENT" is used to initialize the hyperparameters of the algorithm in producing the seed set.

The "EVALUATE" function evaluates the RL agent and the solution population and returns the influence and seed set for IM. Initially an empty seed set array is initialized along with setting influence and total reward as 0. Based on the current state, it computes the Q values. The action corresponding to the highest Q value is chosen after the Q values are sorted in descending order. In the next iterations, the action becomes the state of the algorithm.

It also calculates the reward using the 2-hop IM method and updates the total reward which denotes the total influence of the seed nodes. The tuple {Q_value, State, Action, Reward} is stored in the replay memory for training the RL agent. The function "Eval_all" evaluates all the population and returns the best initial seed set and influence spread to start the algorithm. The initial spread value will act as a fitness evaluator for training the framework for the next iteration.

The function "TRAIN" trains the RL agent and updates its weight parameters to produce results. During training for each batch, some of the random entries are sampled from the replay memory. Each entry is in the form of {Q_value, State, Action, Reward}. During each episode the agent predicts the next state based on the obtained initial fitness spread value. After obtaining the next seed set the reward is calculated by using Bellman equation.

As the iteration progresses, the DQN weights updating is done by back propagation. The loss function is set as Adam optimizer for effective training and preventing the information loss. Then the agent is evaluated using "EVALUATE" function and the RL algorithm is run up to 10 iterations and produces a final seed set.

Followed by the execution of DQN-RL algorithm, the Differential Evolution starts to find the seed set. A random population is generated for the given number of population size, where each member of population is as per the seed set size and has distinct seeds chosen from the network. This random population is then ranked based on the "FastIM" fitness function and the top elite solutions based on its fitness value is remained to be the same and transferred to the next iteration for continuous improvement.

The rest of the solutions are then transferred to the mating pool, where the mutation operation takes place. Post the completion of the mutation step, random crossover of solutions take place. This new set of population is again ranked based on its fitness values obtained. These genetic operations are continued until the termination condition is reached.

As the DQN-RL algorithm, the DE is also run for 10 iterations for each of the DERL epochs. After each iteration, the final results of obtained seed sets from both the algorithms are compared and the best is chosen based on the fitness values obtained. If the DQN-RL achieves a higher fitness value than the DE, then the solution of the DQN-RL is directly appended into the population set of DE. Likewise, if the DE obtains a higher fitness score than the DQN-RL, then the DE seed set is comprised in the population of RL algorithm.

By this adaptive hybridization technique of transferring the fitness values between algorithms, they start to explore new solutions from a more updated and informed solution space, which in turn helps the model to achieve a higher fitness score in small number of iterations.

## 4. Design of experiments

The information on the network datasets selected for the experiment and the parameter values of the DERL framework created to solve the IM Problem in social/complex net-works are provided in the following subsections.

### 4.1 Datasets considered

To evaluate DERL framework and analyze the solution quality alongside the standalone DE and DQN-RL algorithms, seven datasets were chosen carefully from a range of domains and sizes to have an accurate inference from the study. The chosen network datasets consist of those that are uni-partite, directed, and devoid of self-looping.

Table 1. Description of the network datasets used in the experiments

| Name | Type of network | Number of nodes | Node meaning | Number of links | Link meaning |
|---|---|---|---|---|---|
| Dolphins | Community of bottle nose dolphins | 62 | Dolphin | 159 | Association |
| Human protein | Protein Interaction Network | 2,239 | Protein | 6,452 | Interaction |
| Wikipedia | Wikipedia links Network | 2,929 | Article | 1,18,603 | Wiki – link |
| Twitch | Gamers who stream | 7,126 | Streamers | 35,324 | Friendship |
| Cora | Citation Network | 23,166 | Paper | 91,500 | Citation |
| Twitter | Social media Network | 23,370 | User | 33,101 | Follows |
| Google Plus | Social media Network | 23,628 | User | 39,242 | Friendship |

The experimental datasets are thoroughly summarized in Table 1.

### 4.2 Design of DE and DQN-RL in DERL

The adaptive hybrid framework (DERL) uses both DQN-RL and DE to find the possible solutions for seed set that has a high fitness value based on the "FastIM" fitness function, denoting the amount of propagation of information in the network.

As mentioned in the previous section, the DERL is made to run for 10 epochs where each epoch makes the DQN-RL and DE to run for 10 iterations.

Table 2. Parameter settings of the DERL framework

| Parameter | Value |
|---|---|
| Learning rate | 0.001 |
| Gamma | 0.8 |
| Learning times | 10 |
| Buffer Size | 1000 |
| Batch Size | 512 |
| Mutation Rate | 0.01 |
| Crossover Rate | 0.5 |
| Iterations per Epoch | 10 |
| No. of Epochs | 10 |
| Population size of DE and DQN-RL | 100 |
| Seed set size | 10 |

After each epoch, the algorithm's new epoch starts from the previous epoch's state and continues to search for solutions. Table 2 displays the overall parameter value created to effectively address the IM Problem.

Additionally, the DQN-RL algorithm was given an input of a lower dimension of the large network dataset to speed up the process of finding the seed nodes. The net-work dataset was transformed using the LINE (Large-scale Information Network Embedding) embedding method [34].

Line embedding refers to the process of representing individual node data of a network as fixed-dimensional vectors and involves transforming each node data into a numerical representation that captures its semantic and contextual information. Once the textual information is transformed into line embeddings, reinforcement learning algorithms can utilize these representations to learn policies that guide the selection of influential nodes in the network. The rei nforcement learning agent can take actions such as choosing nodes to target, and the rewards it receives are based on the influence or impact achieved through these actions. By iteratively updating its policy based on these rewards, the agent can learn to make better decisions and maximize influence within the network.

By using LINE embedding, the RL agent gains access to a numerical representation that captures the underlying semantics of the text. This enables the agent to better understand the content and context of the textual data, facilitating more informed decision-

making. It also makes the agent to take decisions based on learned representations that capture relevant information about the influence potential of nodes in the network.

## 5. Experimental results and discussion

The DERL framework was created in the aim of achieving higher fitness scores than the standalone DE and DQN-RL algorithms by leveraging the strengths of the Machine Learning and Evolutionary Computation paradigms. An adaptive hybridization technique was employed to have a fitness transfer among the algorithms, making them compensate the weaker solutions generated by the other algorithm's solution.

Seven real world network datasets were chosen and initialized through the "Networkx" python library. A high-performance computing system (HPC) was used to run the DERL framework and the final seed set obtained from the experiment is presented in Table 3.

A sample of obtained plots while performing the experiment that showcases the improvement in solutions over the epochs of DERL framework is shown in Fig. 3, for the twitter dataset.

Further, to evaluate the performance of DERL with DE and DQN-RL, the algorithms were run individually in the same settings as given in Table 2 to solve the IM problem. The Table 4 and Table 5 provide the seed set obtained from the algorithms.

Table 3. Seed set obtained from the DERL framework for the considered datasets

| **Dataset** | **DERL Seed Set** |
|---|---|
| Dolphin | [52, 58, 41, 38, 51, 48, 44, 55, 60, 28] |
| Human Protein | [33, 129, 355, 124, 73, 159, 51, 90, 227, 771] |
| Wikipedia | [588, 580, 587, 1444, 1221, 1256, 1183, 1276, 1255, 448] |
| Twitch | [1773, 792, 581, 4016, 1103, 1089, 1414, 1142, 2186, 1169] |
| Cora | [1477, 490, 5913, 2068, 1478, 1560, 2069, 5843, 3017, 5106] |
| Twitter | [19100, 9621, 4540, 634, 14613, 7746, 825, 6997, 2598, 18799] |
| Google Plus | [8892, 1876, 15599, 2622, 2300, 5958, 4693, 2622, 2376, 267] |

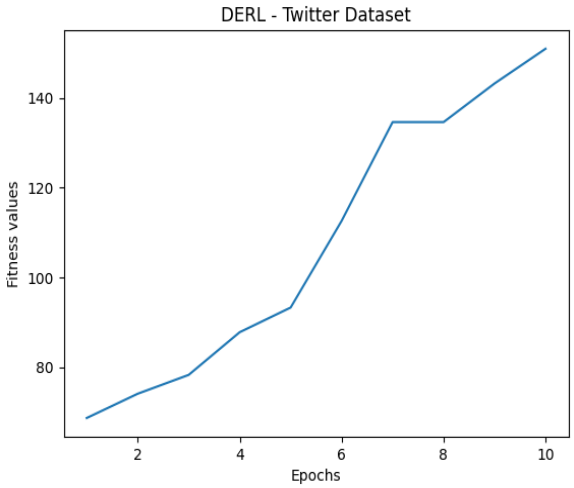


Figure. 3 Epochs vs Fitness value graph for the Twitter dataset

Table 4. Seed set obtained from the DE model for the considered datasets

| Name | DE Seed set |
|---|---|
| Dolphins | [60, 55, 41, 44, 51, 52, 28, 58, 38, 48] |
| Human Protein | [33, 1666, 337, 124, 1546, 355, 129, 771, 73, 442] |
| Wikipedia | [1275, 587, 388, 1276, 1256, 378, 1183, 588, 1083, 580] |
| Twitch | [1360, 792, 2529, 793, 1924, 2887, 2352, 581, 1773, 1883] |
| Cora | [5560, 3157, 490, 2655, 223, 2682, 8125, 1478, 3397, 3236] |
| Twitter | [9014, 4540, 20477, 2251, 14613, 7746, 825, 6997, 2598, 18799] |
| Google plus | [8892, 1876, 15599, 2376, 2300, 5958, 4693, 21152, 4750, 267] |

Table 5. Seed set obtained from the DQN-RL model for the considered datasets

| Name | DQN-RL Seed set |
|---|---|
| Dolphins | [7, 10, 42, 14, 46, 18, 51, 52, 56, 38] |
| Human Protein | [1500, 694, 33, 555, 1045, 1377, 363, 1414, 1624, 573] |
| Wikipedia | [1287, 2516, 2529, 1105, 755, 2737, 946, 588, 1841, 56] |
| Twitch | [287, 4016, 4940, 4330, 1178, 2227, 7052, 5412, 321, 4288] |
| Cora | [3013, 1798, 3017, 490, 2102, 1561, 13212, 4157, 5145, 1407] |
| Twitter | [11824, 6997, 2646, 17609, 897, 18474, 7410, 2488, 10877, 19426] |
| Google plus | [385, 3331, 3118, 2300, 629, 14292, 1726, 7101, 2622, 2711] |

Table 6. Contribution towards best solution in each epoch by DE and DQN-RL

| Name | DE > DQN-RL | DQN-RL > DE |
|---|---|---|
| Dolphins | 9 | 1 |
| Human Protein | 9 | 1 |
| Wikipedia | 6 | 4 |
| Twitch | 9 | 1 |
| Cora | 6 | 4 |
| Twitter | 5 | 5 |
| Google plus | 7 | 3 |

Table 7. Fitness Comparison between DE, DQN-RL, and DERL for the considered datasets

| **Name** | **DE Fitness Values** | **DQN-RL Fitness Values** | **DERL Fitness Values** |
|---|---|---|---|
| Dolphins | 17.4 | 13.5 | **17.43** |
| Human Protein | 195.6 | 48.42 | **215.02** |
| Wikipedia | 1724.7 | 815.42 | **1861.01** |
| Twitch | 321.3 | 64.63 | **330.41** |
| Cora | 80.31 | 58.03 | **85.89** |
| Twitter | 140.77 | 94.26 | **150.93** |
| Google plus | 1077.1 | 699.89 | **1151.85** |

The contribution of fitness transfer of DE and DQN - RL algorithms after each of the epochs of DERL framework is analysed and is presented in Table 6. The values present in Table 6, denotes the number of times the fitness of solution set of one algorithm was greater than the other. It also signifies the contribution of each algorithm in arriving to the final seed set.

As observed from the Table 7, the DERL framework surpasses the standalone DE and DQN - RL algorithms in terms of the final fitness values obtained. DERL framework achieves an average of 155% higher than the fitness scores obtained by DQN-RL algorithm and an average of 6% higher scores than that of DE. Although the scores obtained by the DERL framework is higher, there is a small difference in scores when in comparison with DE. This slightly higher score might be because that the DE is an optimization algorithm and the IM problem is also modelled as an optimization problem whereas the DQN-RL approaches the IM as a learning problem to produce results.

Table 7 represents the comparison of DERL framework with the DE and DQN-RL model based on the fitness values obtained based on their final produced seed set.

Even though there is a small difference in scores between DE and DERL, since these datasets are real-world networks, a higher score might be preferred by advertising recruiters (in terms of marketing) and other applications, where the highest information propagation at the low cost (no. of chosen seed nodes) is necessary.

DE's contribution in producing better results for the smaller datasets (in terms of number of nodes, links) an observed from the Table 6. As the size of the network increases, DQN-RL contributes more to DERL as well.

To compare and evaluate the performance of the proposed framework (DERL) with other existing models, few of the best articles as discussed in Section 2 that uses mathematical modelling, optimization algorithm, Reinforcement Learning Techniques and hybrid frameworks were considered for comparative analysis using a synthetic benchmark dataset.

The benchmark dataset considered is the LFR networks [35], that take node degree and community size heterogeneity into account. Additionally, they enable modelling of real-world networks of any size. To have additional complexity, the control parameter ($\mu$) was varied, whereas $\mu$ increases, the number of links across communities is raised, and the LFR dataset becomes more complex.

With control parameter ($\mu$), the node degree and community size in these networks follow power law distributions. Here, we evaluate the chosen comparison models on two LFR benchmark networks with $\mu = 0.1$ (LFR - A) and $\mu = 0.3$ (LFR - B) for demonstrating how well they perform when dealing with networks that have a variety of node degrees. The parameters for each network are $n = 500$, $\beta = 1$, and $\tau = 2$, and the results are presented in Figs. 4 and 5, for LFR - A and LFR - B datasets, respectively. It was observed that the DERL framework performs better than the existing models, denoting in finding a more effective seed set which propagates information throughout the network than the other models.

Analysis based on Figs. 4 and 5 denotes that when comparing DERL with various mathematical models

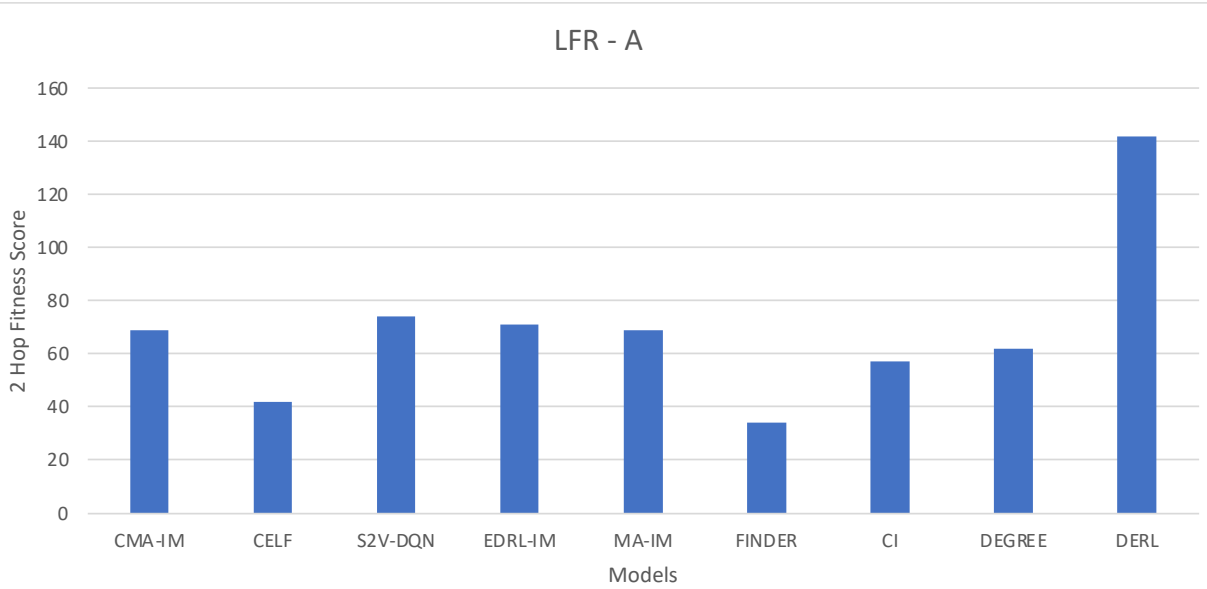


Figure. 4 Comparative Results of DERL framework alongside existing models for LFR – A dataset

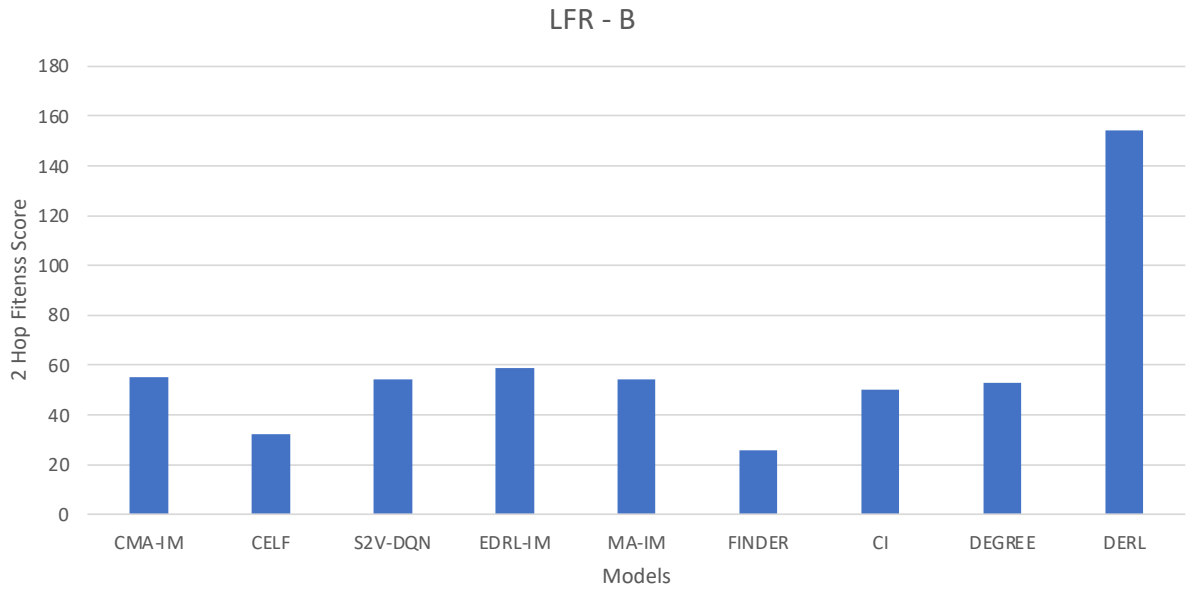


Figure. 5 Comparative Results of DERL framework alongside existing models for LFR – B dataset

used for solving the Influence Maximization (IM) problem such as Degree, CI, CELF, and memetic algorithms like CMA-IM and MA-IM, notable differences in performance emerge.

The DERL framework stands out for its superior performance, achieving a fitness score nearly 2.5 times higher than its counterparts. On the other hand, memetic algorithms like CMA-IM and MA-IM demonstrate similar performance levels, albeit with fitness scores approximately three times lower than DERL.

Among the models examined, the Finder model recorded the lowest fitness score, indicating its relatively limited effectiveness in addressing the IM problem. However, both the S2V-DQN and EDRL-IM models showed slightly better results than other models following DERL.

## 6. Conclusions and future work

This paper introduced the DERL framework, an adaptive hybridization approach that combines Differential Evolution (DE) and DQN-based Reinforcement Learning (RL) to solve the Influence Maximization problem present in social networks. By leveraging the robust optimization, capabilities of DE and the adaptive learning strengths of DQN-RL, the

DERL framework offers an effective solution to the IM problem.

The adaptive hybridization technique ensures that the framework not only converges faster but also produces better performance when compared alongside standalone DE, DQN-RL models, highlighting the potential of combining distinct computational paradigms.

When compared, DERL framework outperforms mathematical models like Degree, CI, and CELF in solving the IM problem, with a fitness score nearly 2.5 times higher. Memetic algorithms like CMA-IM and MA-IM perform similarly but with fitness scores around three times lower than DERL. The hybrid framework of Differential Evolution and DQN - Reinforcement Learning proves most effective, surpassing other models by combining DQN's deep reinforcement learning with Differential Evolution, yielding seed sets with superior propagation potential in networks.

The importance of Evolutionary Learning in solving the IM problem in social networks is highlighted in this paper. Future work will extend the principles of adaptive hybridization to other domains, providing a broader application spectrum for this adaptive hybrid approach.

## Conflicts of Interest

No conflicts of interest to the best of our knowledge.

## Author Contributions

This work represents the findings of the scholars (Author-1,2,3,4) during their colloborative research with equal contribution, under the guidance of mentor (Author-5).

## References

[1] C. L. Streeter and D. F. Gillespie, "Social network analysis", *J. Social Service Res.*, Vol. 16, No. 1-2, pp. 201-222, 1993.

[2] H. L, S, A., G, A., P, S., and G. P. Sajeev, "A Proximity Based Community Detection in Temporal Graphs", In: *Proc. of IEEE CONECCT*, 2020.

[3] A. Uthayasuriyan, G. R. Ramya, and G. Jeyakumar, "Effective Link Prediction in Complex Networks Using Differential Evolution Based Extreme Gradient Boosting Algorithm", In: *Proc. of Advanced Network Technologies and Intelligent Computing: Second International Conference*, ANTIC 2022, Varanasi, India, pp. 149-163, 2023.

[4] G. R. Ramya and P. Bagavathi Sivakumar, "An incremental learning temporal influence model for identifying topical influencers on Twitter dataset", *Social Network Analysis and Mining*, Vol. 11, No. 1, pp. 1-16, 2021.

[5] J. Cheriyan and G. P. Sajeev, "SpreadMax: A Scalable Cascading Model for Influence Maximization in Social Networks", In: *Proc. of 2018 ICACCI*, Bangalore, India, pp. 1290-1296, 2018.

[6] Y. Li, F. Ju, W. Yanhao, and K. L. Tan, "Influence maximization on social graphs: A survey", *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[7] B. Doina, and G. Iacca, "Influence maximization in social networks with genetic algorithms", In: *Proc. of European Conference on the Applications of Evolutionary Computation*, pp. 379-392, 2016.

[8] S. Agarwal, and S. Mehta, "Social Influence Maximization Using Genetic Algorithm with Dynamic Probabilities", In: *Proc. of 2018 Eleventh International Conference on Contemporary Computing (IC3)*, pp. 1-6, 2018.

[9] K. J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network", In: *Proc. of 9th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pp. 137-146, 2003.

[10] F. Moroneand, H. A. Makse, "Influence maximization in complex networks through optimal percolation", *Nature*, Vol. 524, No. 7563, pp. 65-68, 2015.

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

[12] H. Li, R. Zhang, X. Liu, "An efficient discrete differential evolution algorithm based on community structure for influence maximization", *Appl. Intell.*, Vol. 52, No. 1-19, 2022.

[13] 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.

[14] L. Cui, H. Hu, S. Yu, Q. Yan, Z. Ming, Z. Wen, N. Lu, "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.

[15] 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, 2016.

[16] 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, 2021.

[17] A. Uthayasuriyan, H. Chandran, G. K. UV, S. H. Mahitha, G. Jeyakumar, "Performance Evaluation of Evolutionary Algorithms on Solving the Influence Maximization Problem in Social Networks", *International Journal of Modern Education and Computer Science (IJMECS)*, Vol. 16, No. 2, pp. 83-97, 2024.

[18] 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, 2017.

[19] L. Fan, Z. 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.

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

[21] M. Lijia and Z. Shao, and X. Li, and Q. Lin, and J. Li, and L, Victor, C. M, "Influence Maximization in Complex Networks by Using Evolutionary Deep Reinforcement Learning", In: *Proc. of IEEE Transactions on Emerging Topics in Computational Intelligence*, Vol. 7, No. 4, pp. 995-1009, 2023.

[22] H. Li, M. Xu, S. S. Bhowmick, J. S. Rayhan, C. Sun and J. Cui, "PIANO: Influence Maximization Meets Deep Reinforcement Learning", In: *Proc. of IEEE Transactions on Computational Social Systems*, Vol. 10, No. 3, pp. 1288-1300, 2023.

[23] T. Chen, S. Yan, J. Guo, and W. Wu, "ToupleGDD: A Fine-Designed Solution of Influence Maximization by Deep Reinforcement Learning", In: *Proc. of IEEE Transactions on Computational Social Systems*, pp. 1-12, 2023.

[24] W. J. Zhao, L. Li, L. Jiao, J. Liu, and K. Wu, "A Multi-Transformation Evolutionary Framework for Influence Maximization in Social Networks", *IEEE Computational Intelligence Magazine*, Vol. 18, No. 1, pp. 52-67, 2023.

[25] F. Riquelme, P. G. 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.

[26] 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, No. 125584, 2021.

[27] A. Uthayasuriyan, G. H. Chandran, U. V. Kavvin, S. H. Mahitha and G. Jeyakumar, "A Comparative Analysis on Influence Maximization Models in Social Networks", In: *Proc. of 2023 International Conference on Advancement in Computation & Computer Technologies (InCACCT)*, Gharuan, India, pp. 1-6, 2023.

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

[29] H. Singh and T. Singh, "Reinforcement Learning based Search algorithm in Social Network", In: *Proc. of International Conference On Smart Technologies For Smart Nation (SmartTechCon2017)*, Reva University, Bengaluru, 2017.

[30] K. Ali, C. Y. Wang, M. Y. Yeh and Y. S. Chen, "Addressing Competitive Influence Maximization on Unknown Social Network with Deep Reinforcement Learning", In: *Proc. of ASONAM*, The Hague, Netherlands, pp. 196-203, 2020.

[31] J. Uma, P. Vivekanandan, and S. Shankar, "Optimized Intellectual Resource Scheduling using Deep Reinforcement Q-Learning in Cloud Computing", *Transactions on Emerging Telecommunications Technologies*, Vol. 33, No. 5.

[32] S. Rudra and T. S. Kumar, "A Robust Q-Learning and Differential Evolution Based Policy Framework for Key Frame Extraction", *Springer Advances in Intelligent Systems and Computing*, Vol. 1039, pp. 716-728, 2019.

[33] A Uthayasuriyan, G. H. Chandran, U. V. Kavvin, S. H. Mahitha, and G. Jeyakumar, "A Comparative Study on Genetic Algorithm and Reinforcement Learning to Solve the Traveling Salesman Problem", In: *Proc. of MIND 2022, Research Reports on Computer Science*, Issue 1, 2023.

[34] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan and Q. Mei, "Line: Large-scale information network embedding", In; *Proc. of WWW*, pp. 1067-1077, 2015.

[35] 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.