I.J. Modern Education and Computer Science, **2024, 2, 83-97**
Published Online on April 8, 2024 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2024.02.07

Performance Evaluation of Evolutionary Algorithms on Solving the Influence Maximization Problem in Social Networks

Agash Uthayasuriyan
Department of Electrical and Electronics engineering, Amrita School of Engineering, Coimbatore – 641112, India
E-mail: cb.en.u4elc19005@cb.students.amrita.edu
ORCID ID: https://orcid.org/0009-0006-1350-7590

Hema Chandran G
Department of Electrical and Electronics engineering, Amrita School of Engineering, Coimbatore – 641112, India
E-mail: cb.en.u4elc19016@cb.students.amrita.edu
ORCID ID: https://orcid.org/0009-0003-4282-6134

Kavvin UV
Department of Electrical and Electronics engineering, Amrita School of Engineering, Coimbatore – 641112, India
E-mail: cb.en.u4elc19022@cb.students.amrita.edu
ORCID ID: https://orcid.org/0009-0003-7324-7071

Sabbineni Hema Mahitha
Department of Electrical and Electronics engineering, Amrita School of Engineering, Coimbatore – 641112, India
E-mail: cb.en.u4elc19044@cb.students.amrita.edu
ORCID ID: https://orcid.org/0009-0002-1758-0809

Jeyakumar G*
Department of Computer Science and Engineering, Amrita School of Computing, Coimbatore – 641112, India
Email: g_jeyakumar@cb.amrita.edu
ORCID ID: https://orcid.org/0000-0002-5501-2338
*Corresponding Author

Received: 30 April, 2023; Revised: 25 June, 2023; Accepted: 16 July, 2023; Published: 08 April, 2024

Abstract: Influence Maximization (IM) is an optimization problem that deals with identifying the most valuable individuals/ nodes present in the network to attain the maximal information spread when they are activated. Evolutionary Algorithms (EAs) inspired from nature are one of the most powerful methods to solve an optimization problem. This paper attempts to solve the IM problem using few of the popular EAs such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Differential Evolution (DE). These algorithm's performance is evaluated using four comparative metrics, that deal with assessing solution quality, computational efficiency, and scalability. The experimental results of these EAs when tested on several real-world networks reveal that the GE and PSO algorithms were found to produce relatively poorer quality of seed nodes and also with higher computational costs, making it less preferrable. DE was able to obtain the best seed sets (in terms of influence spread value) than other algorithms and ACO, the fastest among all the considered algorithms in terms of execution time, was found to obtain seed set with appreciable influence spread with a slightly higher computational cost than DE.

Index Terms: Social network, Influence Maximization, Seed nodes, Evolutionary Algorithm, Genetic Algorithm, Ant Colony Optimization, Particle Swarm Optimization, Differential Evolution.

1. Introduction

Social Network Analysis (SNA) deals with the study of relationships between individuals, groups, or organizations [1] using tools based on mathematical, Machine learning, and Artificial intelligence models. These networks are represented in the form of graphs, where each individual/ item is represented as nodes and the interconnections between them are represented as links. A few of the most researched topics in SNA are Link Prediction [2], Community Detection [3], and Influence Maximization.

The Influence Maximization (IM) problem originates from the need to find the key nodes/ individuals present in the network, who acts as a catalyst in terms of spreading information such that when these nodes are activated (e.g., by being recruited or shown an advertisement) will lead to the largest number of nodes being influenced. The solution to this IM problem has important applications in various fields, such as marketing, public health, and social activism.

For instance, in marketing, businesses want to find a select group of very powerful clients who can aid in spreading the word about their goods or services to a larger audience. In the field of public health, recognizing important people who are more susceptible to illness can aid in preventing the spread of diseases. In social activism, finding influential persons who can promote social change can aid in mobilizing bigger crowds and achieving objectives. Additionally, solving the IM problem can also be used in finding the rumor spreader in social media platforms, identifying the masterminds behind terrorist networks, research work that led to many major inventions, and also in a biological setup like protein networks. The influence maximization problem is concerned in all of these applications with identifying a small fraction of individuals (seed set) who have the greatest impact on the transmission of information, ideas, or behaviors in a social network. However, the challenging task is to find a good quality seed set (set of nodes that has the maximum amount of influence in the network when activated) in a short time and at lower computations.

Although the IM can be solved through many techniques [4, 5], since it is an optimization problem, it is most suitable to solve using EAs. EAs inspired from Darwin's principle of "Survival of the fittest", have proved to solve problems dealing with single/multi-objective functions under constrained/ unconstrained conditions. These algorithms that will run for several iterations, learn from the solutions of the previous iteration and improve them continuously to attain the maximum/ minimum of the designed objective function.

This paper aims to provide solution for the Influence Maximization problem present in social networks using a few of the most popular EAs – GA, ACO, PSO, and DE. These mentioned EAs are designed to solve the IM problem and are compared based on four comparative metrics to study the performance, strengths, and weaknesses of each of the algorithms. These comparative metrics were carefully chosen based on the stochastic nature of algorithms, and also to test the solution quality, computational efficiency, and scalability of the EAs.

The remaining parts of the paper is organized as follows. In section 2, the recent and significant research work related to solving the IM problem has been reviewed. Section 3 demonstrates the detailed designing technique followed for the algorithms along with other experimental settings. Section 4 provides the obtained experimental results with related discussions and finally, Section 5 concludes the paper.

2. Background Study

The recent and most notable studies that discuss about solving the influence maximization problem in social networks relating to mathematical, evolutionary techniques have been studied to understand the nature of the problem and summarized as follows.

To overcome the greedy technique's high computing cost and restricted spreading potential, the authors in paper [5] proposed an innovative approach titled SpreadMax, which involves two stages, where seed nodes are selected utilizing a hierarchical accessibility method in the initial process and the above-chosen seed nodes propagate in the later stage. This approach is experimented with real-world data in contrast with conventional methods and is noticed that the SpreadMax method outperforms others in terms of maximizing the propagating process and minimizing computational overhead. Influenced by the potential for unstable channels of interaction, the authors developed an effective influence assessment approach dependent upon total value as well as change in assessment of adjacent nodes [6]. They subsequently built a discrete moth-flame optimization (DMFO) approach and applied it to IM, taking advantage of regional topology of networks to effectively locate node combinations that have the most significant influence. They further provided a search area selection method that is adjustable to the seed set size to better the speed of the algorithms convergence. The experimental findings on five real-world peer networks, assessing the proposed approach with various possibilities in the existing literature show that it is efficient and robust when solving the influence maximization problem.

In the paper [7], the authors have presented an evolutionary technique named degree-descending search evolution (DDSE) to solve influence maximization problem. By avoiding repetitive computations with an estimation function EDV, DDSE solves the inefficient behaviour of greedy models. The DDSE is built using degree-based algorithm named degree descending search, and expertly designed operators such as initialization, mutation, crossover, and greedy selection. DDSE is about five times quicker compared to the cutting-edge greedy technique while preserving a high standard of precision, under results from examination of practical networks. The authors in paper [8] have suggested

IICEA, a confined evolutionary algorithm that utilizes local-global influence indicators, for resolving the budgeted influence maximization problem with great efficacy. Two aspects are taken into account in IICEA: local neighbour and global community data, which can be used to more accurately evaluate the impact of nodes in social networks.

An innovative agent-based evolutionary method (ABEM), has been presented by the authors of the paper [9] for mining influencers in social networks on the internet. This method optimizes the choice of the influencer set in a dynamic setting while recognizing the users' ability to have an impact in a dispersed way. The results of the experiment show that ABEM does not just beat current algorithms in influence maximization, but additionally also have the potential for use in a huge-scale, changing circumstances.

To enhance the efficiency of evolutionary-based influence maximization methods and prohibit users from making a substitute model selection, paper [10] introduces a multi-transformation evolutionary framework (MTEFIM) for IM. It makes use of the commonalities and distinct benefits of various proxy models. Every transition is given a population. Amount of similarity across the people who have undergone various transformations gives an indirect indication of the possible relationships between them. A knowledge-sharing mechanism between a specific transformation and the majority pertinent "assisted" alteration is created to circulate shared information adaptively depending on the inter-transformation connection. Finally, the proposed model analyses the detailed rank of the ideal set of seeds from every transformation and produces an ultimate seed set incorporating the results from all substitute versions.

The authors of [11] presented a probability-based multi-hop diffusion method that addresses the spread maximisation problem in social networks to overcome the limitations of greedy and heuristic-based techniques. While maintaining an identical fast computation rate as other algorithmic methods, it can accomplish greater influence diffusion through two significant changes. First, the impact of both direct and multi-hop neighbours is looked at. Second, the variety of propagation possibilities within various pairs of nodes is considered, resulting in improved performance. Experiments are carried out using two real-world data sets to analyze and contrast the outcomes of the suggested algorithm and its alternatives. According to the findings, the proposed algorithm consistently exceeds the alternative by 6-16%.

For solving the influence maximization problem, the authors have opted the gray wolf optimization algorithm as a population-driven optimizing method in the paper [12]. However, it has some constraints, which are primarily due to the use of a population-based approach. Despite its efforts to avoid local minima, it might still get trapped in one, implying that the ultimate solution will result in local minima instead of global minima.

After studying these previous researches, this paper is aimed at solving the Influence Maximization problem in Social Networks utilising EAs such as GA, ACO, PSO, and DE along with a performance evaluation between each of them.

3. Design of Experiments

EAs, a part of the Evolutionary Computation (EC) paradigm, are a family of computational techniques inspired by the biological process of natural selection. EAs are used to solve optimization and search problems [13,14] by evolving a population of candidate solutions through successive generations. In each generation, the fittest individuals are selected to reproduce and produce offspring, which are then subjected to mutation and recombination to introduce variation into the population. Over time, the population converges towards better and better solutions to the problem provided.

This paper discusses a few of the most popular EAs – Genetic Algorithm, Ant Colony Optimization, Particle Swarm Optimization, and Differential Evolution in solving IM problems under similar, comparable experimental setups to draw results on the performance of each of the algorithms. These algorithms have also individually exhibited to be powerful in solving a variety of single and multi-objective optimization problems. In solving IM problem, the algorithms start with a random initial set of nodes chosen from the network (known as initial seed set) and keeps on improving them using the fitness/ influence spread value obtained. This process is continued until termination condition and the best seed set from the overall dataset is found by the algorithms.

To find the influential nodes present in the network using EAs, there is a need of an influence propagation model, that will be used to evaluate the fitness of the produced seed set. The Linear Threshold (LT) [15], Independent Cascade (IC) [16], and other diffusion models [17] are some examples of influence propagation models. This paper focuses on using the IC model, and the work can be extended to other models as well.

The "word-of-mouth" impacts, or how people can be influenced by the beliefs, choices, and actions of their friends and neighbours, are the driving force behind the IC model. The IC model's core principle is that each node, once engaged, has the capacity to activate its neighbours based on the edge propagation probability, and that this propagation occurs independently and recursively until no other node is activated.

Performing the standard IC propagation steps on a large-scale network takes time since the final propagation step (PS) at the termination phase may be quite large and it is difficult to predict the final PS value before execution. To overcome this, Lee and Chung have suggested a fast approximation IC model [18], which limits the influence ability of seed nodes to nodes within a 2-hop range of them (such as neighbours' neighbours), in order to streamline the propagation processes. The final PS value in this instance is no more than 2. Several analysis and tests in their research have demonstrated that the fast approximation IC model (FastIM) is able to effectively assess the influence spread of

the provided seed set. Therefore, FastIM has been chosen as the fitness function, as it is quicker and also produces reliable results.

The following subsection provides a brief on each of the mentioned EAs along with their respective design methodology in solving the IM problem in social networks.

3.1. Genetic Algorithm

GA, is one of the most researched and widely used EA [19], used for several optimization tasks. The algorithm mimics the process of evolution in nature where the fittest individuals are more likely to survive, and the idea that a population of solutions can be improved by selectively breeding the best individuals. The GA applies a set of operations to the population of solutions that include initialization of the population, calculation of fitness value, followed by selection, crossover, and mutation. The detailed working methodology of GA is presented in Fig. 1.

Algorithm 1: Genetic Algorithm
Input*: Graph, population size, elite size, generations, mutation rate, generations, seed set size*
Output*: Seed set, Influence Score*

1. Initialize Graph
2. Set Fitness = FastIM (seed nodes)
3. Set M = Mutation rate
4. Define Function Crossover (mating pool, seed set size):
    a. Pop = [ ]
    b. While length (Pop)! = length (Mating pool):
        i. a, b = random solutions in mating pool
        ii. crossover point = random integer in range [0, seed set size]
        iii. child = a [0, crossover point] + b [crossover point, length(b)]
        iv. Pop. Append(child)
    c. Return Pop
5. Define Function Mutation (C_Population, M):
    a. Pop = [ ]
    b. While length (Pop)! = length (C_Population)
        i. i = 0
        ii. Pos = random integer in range [0, seed set size]
        iii. Value = Random number
        iv. If Value < M:
            1. Temp = Replace C_population[Pos] with random node
            2. Pop. Append (Temp)
        v. i+=1
    c. Return Pop
6. Initialize population as per given population size and seed set size
7. Rank the population based on fitness results
8. Set Best solution = 1st solution in ranking
9. For i in range of generations:
    a. Select the top solutions as per given elitism value
    b. Transfer the remaining solutions to mating pool
    c. C_Population = Crossover (mating pool, seed set size)
    d. M_Population = Mutation (C_Population, M)
    e. Population = M_Population
    f. Rank the population based on fitness results
    g. If fitness of (1st solution in ranking) > fitness of (Best solution):
        i. Best solution = 1st solution in ranking
10. Return Best solution, fitness (Best solution)

Fig. 1. Design of Genetic Algorithm

Initialization of Population and Fitness Calculation: In solving the IM problem, the GA starts with a random initial population, where each solution of the population is a set of seed nodes. The fitness value of each solution in the population is evaluated by measuring the influence spread of the selected seed nodes in the network using the "FastIM" model. The score obtained from the fitness function is proportional to the influence spread it generates.

Selection: It is used to choose the fittest individuals in the population for reproduction. In this implementation, a simple ranking method is used to sort the solutions in descending order of their fitness scores. The top-ranked solutions based on elitism value are retained, while the others are transferred to the mating pool for crossover and mutation.

Crossover: Crossover is the process of combining two parent solutions to create a new offspring solution. In this implementation, a crossover operator is used that selects a random subset of seed nodes from each parent and combines them to create a new solution, as shown in Fig. 2.

Solution 1	23	1	4	52	17	35	46	57	44	3
Solution 2	49	58	7	33	29	40	51	36	8	1
Crossovered Solution	23	1	4	52	17	40	51	36	8	1

Fig. 2. A Sample of GA Crossover

Mutation: Followed by crossing over of solutions, mutation takes place where a random node of the solution is replaced with another node. The Mutation is the process of introducing random changes in the offspring solution to promote diversity in the population. In this implementation, a mutation operator is used that randomly swaps two seed nodes in the solution with the given mutation factor. This acts as an important function in GA by introducing novel seed nodes and ruling out the possibility of local convergence.

The new population is considered as the current population and this process continues until the termination criteria are met.

3.2. Ant Colony Optimization

ACO, a metaheuristic algorithm is based on the behavior of ants in their search for food. In ACO, artificial ants are used to find the optimal path in a graph or network. When ants search for food, they leave a trail of pheromones that other ants can follow. The strength of the pheromone trail depends on the amount of food found. Similarly, in ACO, artificial ants leave pheromone trails on the edges of the graph they traverse. The strength of the pheromone trail is proportional to the quality of the solution found.

The algorithm works by iteratively constructing solutions using artificial ants. In each iteration, each ant builds a solution by selecting edges based on a probabilistic rule that depends on the pheromone trail strength and the distance between nodes. After all, ants have completed their construction, the pheromone trail is updated to reflect the quality of the solutions found. The stronger the solution, the higher the pheromone trail strength. In solving the IM problem, the ACO algorithm has been designed based on the following steps.

Ant Generation: A set of ants is generated that start from the current seed nodes and explore the network to find high-quality nodes based on a combination of pheromone trail and heuristic information. Specifically, each ant starts from one of the current seed nodes and iteratively selects a new node to visit based on a combination of two factors: the pheromone amount on the edge that connects the current node and the potential next node, and a heuristic estimate of the quality of the potential next node. The ant continues this process until it reaches the required number of seed nodes.

Seed Node Selection: The new seed nodes are chosen based on the updated pheromone trail and the quality of the nodes found by the ants. A probability distribution is calculated over the nodes in the network, based on the pheromone level on the edges connecting the current seed nodes to the potential new nodes, the heuristic information and the parameters alpha, and beta. A new seed node from this distribution of possible seed nodes is then selected randomly.

Pheromone Update: Updating of the pheromone trail is done based on the fitness value of the nodes found by the ants, with higher quality nodes receiving a higher amount of pheromone. The updating of the pheromone level on the edge connecting each visited node to the following node visited by the ant is done based on the quality of the nodes visited by the ant. Higher-quality nodes receive a higher amount of pheromone, which reinforces the pheromone trail on the edges leading to those nodes.

Evaporation: The pheromone trail over time is evaporated to avoid local optima and encourage exploration. The level of pheromone on all edges of the network is reduced by a fixed evaporation rate at each iteration, which models the decay of pheromone over time. This encourages the ants to explore new parts of the network and avoid getting stuck in local optima.

Overall, these four steps of the ACO algorithm work together to identify a set of high-quality seed nodes for influence maximization, by iteratively updating the pheromone trail, selecting new seed nodes, and exploring the network using the behavior of ants. The working methodology of the Ant Colony Optimization Algorithm is presented in Fig. 3.

Algorithm 2: Ant Colony Optimization

Input: *Graph, seed set size, number of ants, Alpha, Beta, Evaporation rate, pheromone levels, number of seed nodes*

Output: *Seed set, Influence Score*

1. Initialize Graph
2. Initialize Candidate solution = ϕ
3. Fitness Function = FastIM (seed nodes)
4. For each ant in the number of ants:
    a. Choose a random starting node
    b. Build the candidate solution using ACO decision rule
        i. Get list of neighbors of current node
        ii. If no neighbors: Return random node
        iii. heuristic_info = [(v, len (set (Graph. neighbors(v))) - len (set (Graph. neighbors(v)) & set(candidate_solution))) for v in neighbors]
        iv. Set ProbabilityList= [ ]
        v. For node in list(neighbors):
            1. Get pheromone level of edge connecting current node and neighbor.
            2. Probability = (pheromone_level^alpha) *(heuristic_info^beta).
            3. Append Probability to ProbabilityList.
        vi. Total_probability = Sum (ProbabilityList)
        vii. If total probability = 0, then select random neighbor
    c. Calculate fitness of the candidate solution
    d. If fitness (candidate solution) > fitness (Best_candidate_solution),
        i. Set Best_candidate_solution = Candidate solution
5. For each edge:
    a. Update the pheromone level (Pheromone_level (i, j) = Pheromone_level (i, j) *(1 - evaporation_rate) + (evaporation_rate*fitness_value)
6. Return the Best candidate solution, Fitness (Best candidate solution)

Fig. 3. Design of Ant Colony Optimization Algorithm

3.3. Particle Swarm Optimization

The behaviour of fish schools or flocks of birds serves as the basis for PSO. In PSO, a population of particles moves in a multi-dimensional search space to find the optimal solution, where each particle has a separate position and velocity, which are updated iteratively based on the best-known position and solution of the whole swarm.

In the context of the IM problem, PSO is used to identify seed nodes in a network that has the maximal spread of influence by iteratively updating the positions of the particles (i.e., potential seed sets) and evaluating their fitness through fitness function. After each iteration, the best seed sets are selected as the current best positions for each particle, and the best seed set among all particles is selected as the global best position.

The algorithm begins by initializing an empty particle seeds list, and for each particle, a new velocity is computed based on its personal best position, current velocity, and global best position. The parameter “w” is used to balance the trade-off between exploration (i.e., the particle's velocity changes to search for new solutions) and exploitation (i.e., the particle moves towards the best solution found so far). If w is too large, the particles will explore too much and may not converge to a good solution, whereas when w is too small, the particles will converge too quickly and might get converge in local optima.

The Fast IM method is then used to assess the fitness of each particle's seed set, and if the current fitness is higher than the particle's personal best fitness, the position and fitness are changed accordingly. The global best position and fitness are updated if the maximum personal best fitness value is higher than the global best fitness value. Finally, the best seed set and its corresponding influence score are returned.

The tendency of the particle to travel towards its individual best position and the global best position are controlled by the parameters "C1" and "C2," respectively. Fig. 4 presents the application of these parameters and the PSO algorithm's operational technique.

Algorithm 3: Particle Swarm Optimization

Input: Graph, num_particles, num_iterations, w, c1, c2, seed_set_size
Output: Best seed set, Best influence score

1. Initialize Graph
2. Set Fitness function = 2hop(seed nodes)
3. Set particles = random population for given num_particles size
4. Calculate fitness of each particle, consider it as it's personal_best_fitness and personal_best_position as the present position of particles
5. Set Global best fitness = max(all particle's fitness)
6. Set Global best position = particles[position of max. fitness value]

7. For i in range(num_iterations):
    a. Set particle_seeds=[ ]
    b. For j in range(num_particles):
        i. Set r1, r2 = randomly filled arrays with size of no. of. nodes present in the graph
        ii. Set velocities = w * particles[j] + c1 * r1 * (personal best[j] – particles[j]) + c2 * r2 * (global best position – particles[j])
        iii. Particles[j] = velocities
        iv. Set top values of particles[j] until seed_set_size as 1.
        v. Set remaining values of particles[j] as 0.
        vi. Particle_seeds.append (Indices of 1 in particles[j])
    c. Current_fitness = Fitness function(each) for each in particle_seeds
    d. For j in range(num_particles):
        i. If Current_fitness[j] > Personal_best_fitness[j]:
            1. Personal_best_position = particles[j]
            2. Personal_best_fitness = Current_fitness[j]
    e. If max(Personal_best_fitness) > Global best fitness:
        i. Set Global best position = particles [position of max. fitness value]
        ii. Set Global best fitness = max(Best fitness value)

8. Best_seed_set = Indices of 1 in Global best position
9. Best_influence_score = Global best fitness
10. Return Best_seed_set, Best_influence_score

Fig. 4. Design of Particle Swarm Optimization Algorithm

3.4. Differential Evolution

DE, the popular and powerful stochastic optimization algorithms of the EAs family uses a unique mutation operation among other EAs, in the creation of an offspring population. In DE, each solution is represented as a vector of real-valued parameters and uses the differential information between solutions in the population to guide the search towards better solutions. The mutation and recombination steps introduce diversity into the population, while the selection step ensures that the best solutions are preserved and improved upon in the next generation. The global best position and fitness are updated if the maximum personal best fitness value is higher than the global best fitness value. Finally, the best seed set and its corresponding influence score are returned.

In solving the IM problem, the DE works similarly to GE but with a difference in the order of execution and mutation operation. The working methodology of DE in solving the IM problem in social networks is presented in Fig. 5.

Algorithm 4: Differential Evolution
Input*: Graph, population size, seed set size, Generations, elite size, Mutation rate, Crossover rate*
Output*: Seed set, Influence Score*

1. Initialize Graph
2. Set Fitness = FastIM (seed nodes)
3. Initialize Population as per given population size and seed set size
4. Rank the population based on fitness function results
5. Set Best solution = 1st solution in ranking
6. Set Elite = top elitism value number of solutions as per given input
7. Set F = Mutation rate; CR = Crossover rate
8. Define Function Mutation (Population, seed set size, F):
    a. Initialize new population = []
    b. While length (new population)! = length (population):
        i. a, b, c = random solutions in population
        ii. for i in range (seed set size):
            1. v[i] = a[i] + F * (b[i] – c[i])
            2. v[i] = round(v[i])
        iii. new population. append (v)
    c. Return new population
9. Define Function Crossover (Population, Mutated population, CR):
    a. Initialize new population = []
    b. While length (new population)! = length (population):
        i. i = 0
        ii. Probability = random number in range [0,1]
        iii. If Probability < = CR:

u = Mutated population [i]
Else: u = Population [i]

        iv. new population. append(u)
        v. i+=1
    c. Return new population
10. For i in range of generations:
    a. Mutated population = Mutation (Population, seed set size, F)
    b. Crossed population = Crossover (Population, Mutated population, CR)
    c. Rank crossed population based on Fitness function
    d. Replace bottom elitism value of solutions with Elite
    e. If fitness of (1st solution in ranking) > fitness of (Best solution):
        i. Best solution = 1st solution in ranking
    f. Update Elite = top elitism value number of solutions

Return Best solution, fitness (Best solution)

Fig. 5. Design of Differential Evolution Algorithm

The population (set of seed nodes) is initially chosen at random by the algorithm, following which mutation and crossover occur. The standard DE/rand/1 operation, which is provided in Eq. (1), has been followed, that selects three random solutions (vectors) from the population and calculates a new vector by scaling the difference of two vectors. There is a presence of scaling factor "F", that controls the amount of difference to be added. A higher value of "F" leads to more exploration of the solution space, while a lower value of "F" leads to more exploitation of the current solutions.

$$Mutated\ vector = X1 + F(X2 - X3) \tag{1}$$

The new solution is ensured to be unique and is then made to crossover. The crossover process is a key component of the DE algorithm, and it is used to combine the information from two candidate solutions to create a new candidate solution. The presence of crossover rate "CR" controls the probability of choosing a part of the solution from the mutated solution, which allows for exploration of the solution space. The resulting solution is then added to the new population, as depicted in Eq. (2).

$$\text{Vector} = \begin{cases} \text{Mutated Vector if Random Probability} \leq \text{CR}; \\ \\ \text{Existing Vector if Random Probability} > \text{CR}; \end{cases} \tag{2}$$

Followed by mutation and crossover of the whole population, each of the solutions in the population is ranked based on the value obtained from the fitness function "FastIM", after which the top solution is chosen as the particular iteration's best solution. If the particular iteration's best solution is better than the overall best solution across all iterations, then the overall best solution is updated accordingly.

It is also made sure that the top solutions as per chosen elitism value are transferred to the population in the next iteration, to make sure that the best solutions exist and the algorithm does not deviate to search for other poor solutions. This process is repeated continuously until the termination criteria are met.

3.5. Design of parameters for EAs and Datasets considered

To compare and evaluate the discussed EAs on a fair scale, each of the algorithms was designed to deal with a maximum of 10000 solutions throughout their execution. i.e., The GA and DE start with 100 solutions and runs for 100 iterations, while the PSO and ACO were made to start with 1000 particles/ ants, running for 10 iterations. The overall values of parameters involved in each of the algorithms are presented in Table 1. Additionally, the research works [20, 21, 22] were studied to attain the best results for each of the algorithms.

Table 1. Design of Parameters

Algorithm	Parameters
Genetic Algorithm	Population size = 100, Generations = 100, Mutation rate (M) = 0.01, Elite Size = 10%
Ant Colony Optimization	Number of ants = 1000, Number of iterations = 10, Evaporation Rate = 0.1, Alpha = 2, Beta = 2
Particle Swarm Optimization	Number of particles = 1000, Number of iterations =10, W = 0.5, C1 = 1, C2 = 1
Differential Evolution	Population size = 100, Generations = 100, Mutation rate (F) = 0.01, Crossover rate = 0.5, Elite Size = 10%

These algorithms were then tested using eight distinct real-world datasets gathered from the KONECT project [23], which were chosen from a range of domains to generalize the comparison task. The networks in the study consist of those that are uni-partite, directed, and devoid of self-looping. Table 2 provides a thorough summary of the experimental datasets.

Table 2. Description of Considered datasets

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
Epinions	Trust Network	75,879	User	5,08,837	Trust

4. Results and Discussion

4.1. Experimental Results

The eight real-world datasets that will be used for testing the algorithms were initialized through the "Networkx" python library. The specifics of the system used to perform experiments mentioned in the paper are as follows: Apple MacBook pro with 2.0GHz quad‑core 10th‑generation Intel Core i5 processor, 16GB of 3733MHz LPDDR4X RAM, and 1TB SSD.

All the algorithms were executed for 10 independent runs, and the best run (consisting of the highest fitness score) has been considered. Table 3 presents the best seed set obtained for each of the discussed EAs for the mentioned datasets.

Table 3. Best seed set obtained by EAs across all iterations

Name	GA Seed set	ACO Seed Set	PSO Seed set	DE Seed set
Dolphins	[55, 41, 52, 48, 58, 44, 45, 28, 46, 53]	[7, 10, 42, 14, 46, 18, 51, 52, 56, 38]	[43, 46, 47, 51, 12, 53, 56, 49, 60, 62]	[60, 55, 41, 44, 51, 52, 28, 58, 38, 48]
Human Protein	[129, 573, 90, 1735, 5, 325, 95, 473, 203, 2053]	[129, 227, 996, 1063, 8, 168, 1866, 10, 267, 376]	[34, 74, 130, 147, 204, 327, 356, 772, 1297, 1664]	[33, 1666, 337, 124, 1546, 355, 129, 771, 73, 442]
Wikipedia	[1231, 422, 428, 891, 931, 1281, 434, 1205, 790, 1444]	[1224, 587, 588, 593, 1108, 665, 1210, 1246, 1182, 1255]	[397, 439, 449, 589, 671, 712, 758, 791, 1192, 1277]	[1275, 587, 388, 1276, 1256, 378, 1183, 588, 1083, 580]
Twitch	[3566, 569, 581, 124, 3320, 2447, 675, 6055, 798, 6127]	[6563, 1924, 1773, 1869, 1072, 4949, 3285, 5913, 1883, 4347]	[1277, 1414, 3118, 4913, 3185, 971, 3305, 217, 5161, 5216]	[1360, 792, 2529, 793, 1924, 2887, 2352, 581, 1773, 1883]
Cora	[7830, 6774, 5913, 5464, 6395, 884, 2773, 2718, 12860, 4157]	[3013, 1798, 3017, 490, 2102, 1561, 13212, 4157, 5145, 1407]	[991, 1562, 2882, 2941, 3117, 6865, 7310, 7379, 15299, 17697]	[5560, 3157, 490, 2655, 223, 2682, 8125, 1478, 3397, 3236]
Twitter	[11824, 6997, 2646, 17609, 897, 18474, 7410, 2488, 10877, 19426]	[1924, 228, 844, 6509, 1773, 2447, 4591, 3285, 4665, 4949]	[826, 2680, 2718, 5591, 10181, 10406, 13703, 14486, 20123, 20871]	[9014, 4540, 20477, 2251, 14613, 7746, 825, 6997, 2598, 18799]
Google plus	[12108, 3586, 12478, 11059, 23562, 3879, 17312, 14583, 18440, 9361]	[3937, 2342, 208, 3281, 1876, 2300, 4060, 7101, 2622, 3600]	[2009, 2223, 4804, 8551, 13696, 14528, 15051, 15600, 22469, 23210]	[8892, 1876, 15599, 2376, 2300, 5958, 4693, 21152, 4750, 267]

A sample of obtained plots while performing the experiment that showcases the improvement in solutions over the iterations for all the EAs is shown in Fig. 6, for the twitter dataset.

4.2. Comparative Metrics for Performance evaluation

To conduct a performance evaluation of the considered EAs, four comparative metrics have been chosen from the literature of Evolutionary Computing. These metrics act as a tool to statistically evaluate the algorithms based on the outputs obtained from the considered datasets. The description of the metrics and their recorded results are as follows.

Fitness score (FS): As discussed in the earlier parts of the paper, the fitness function of the entire experiment has been kept to be the "FastIM" model. The output of the fitness function is used as a measure of solution quality and is preferred to have a higher magnitude, denoting a better seed set. Table 4 presents the best FS values recorded for each of the EAs.

System Execution Time (SET): The system running time of each of the algorithms from its first step to the completion of the entire run is calculated for each of the EAs. This metric, expressed in seconds is preferred to have a lower magnitude, denoting faster execution of the algorithm. The lesser the execution time, the more will be the scalability of the EA. The SET of all the EAs in identifying the best seed set is presented in Table 5.

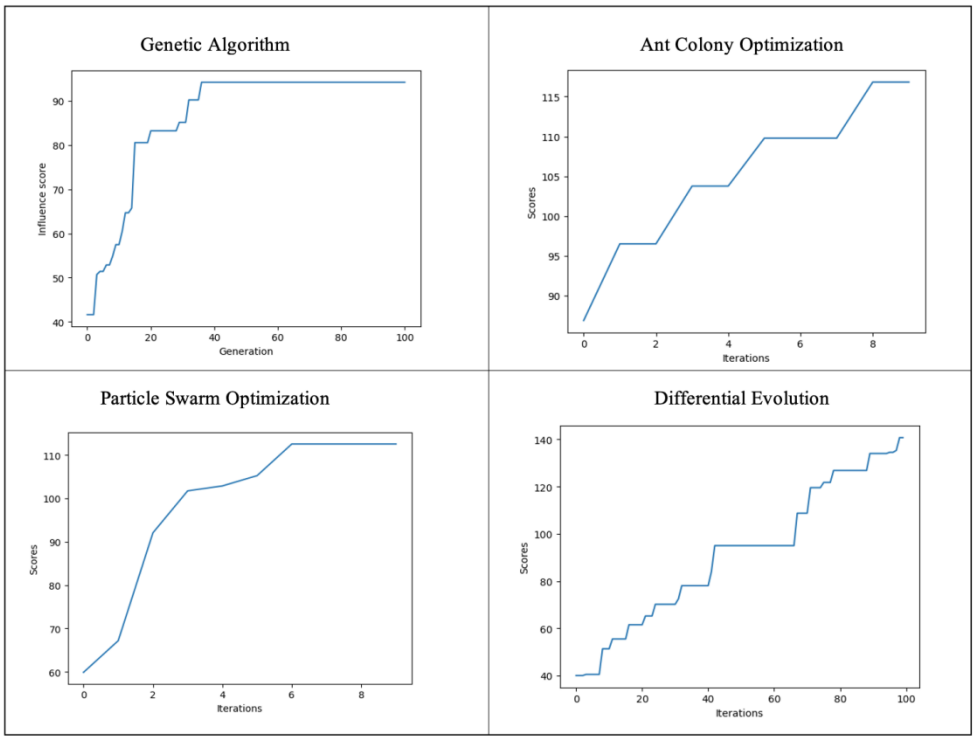


Fig. 6. EAs Fitness score vs Iterations graph of Twitter dataset

Table 4. Best FS values across all iterations of EAs wrt. considered datasets

Name	GA - FS	ACO - FS	PSO - FS	DE - FS
Dolphins	17.32	13.59	17.2	17.5
Human Protein	82.8	90.38	146.43	195.6
Wikipedia	985.2	1355.9	1372	1724.7
Twitch	154.9	211.27	269.1	321.3
Cora	43.1	58.03	48.5	80.31
Twitter	94.26	116.83	112.5	140.77
Google plus	113.7	980.04	260.3	1077.1
Epinions	300.92	5717.97	1718.9	4464.7
Normalised Avg. Score	0.48	0.77	0.66	0.97

Table 5. SET scores of all EAs wrt. considered datasets

Name	GA – SET (seconds)	ACO – SET (seconds)	PSO – SET (seconds)	DE – SET (seconds)
Dolphins	10.48	0.68	7.47	11.96
Human Protein	48.4	10.3	85.9	128.7
Wikipedia	523.7	339.4	431.4	911.46
Twitch	108.08	39.2	171.7	126.2
Cora	30.5	15.13	112.6	47.68
Twitter	53.6	9.47	130.6	31.9
Google plus	65.9	15.02	174.7	662.4
Epinions	438.3	1538.04	1095.9	1976.4
Normalised Avg. SET	4.59	1.31	7.54	11.39

For the calculation of the forthcoming metrics, “Minimum objective function value (Mof)” shall be used, which is the minimum of the best influence spread obtained from all the algorithms. This Mof shall be set as a threshold, as shown in Table 6 and the discussed EAs will be tested accordingly.

Table 6. Mof values of each of the considered datasets

Name	Mof
Dolphins	13.59
Human Protein	82.8
Wikipedia	985.2
Twitch	154.9
Cora	43.1
Twitter	94.26
Google plus	113.7
Epinions	300.92

Based on the FS scores obtained by the algorithms for each of the considered datasets as shown in Table 4, the minimum fitness scores obtained for each dataset is chosen as the MOF value. Since a minimum value is chosen as MOF, it can used to test the algorithms, by having it as a threshold value.

Success Rate (SR): The success rate is calculated by dividing the number of successful runs by the total number of runs, where a run is considered successful when the Mof is reached before the maximum number of generations (total runs). The algorithms were put to 10 runs for each of the datasets, and the successful runs were then observed and recorded. Table 7 presents the SR rates, where a higher magnitude of Avg. SR is considered to be good than a lower one.

Convergence Measure (CM): This metric is calculated as the number of fitness function evaluations made to reach the “Mof” and is normalized by dividing the result by the maximum number of fitness evaluations possible for the algorithm to make in its entire run. This parameter expressed in percentage signifies the computational efficiency of the algorithm in terms of accessing the fitness function. The EA that uses the fitness function the least number of times, yet reaches the Mof is considered to be an efficient algorithm. The results of calculating this parameter are presented in Table 8.

Table 7. SR scores of all EAs wrt. considered datasets

Name	GA SR	ACO SR	PSO SR	DE SR
Dolphins	10	4	10	10
Human Protein	6	10	8	10
Wikipedia	9	10	9	10
Twitch	6	10	10	10
Cora	4	10	8	10
Twitter	1	9	5	10
Google plus	8	10	10	10
Epinions	8	10	10	10
Avg. SR	6.50	9.13	8.75	10.00

4.3. Analysis of obtained results

Influence Maximization problem present in social networks [24, 25] is NP-hard, and there is no definitive perfect solution set for all the datasets considered. Therefore, the comparative metrics – Fitness Score (FS), System Execution Time (SET), Success Rate (SR), and Convergence Measure (CM) deal with assessing the solution quality, scalability, performance, and robustness of models which will be helpful in the evaluation of the performance of EAs - GA, ACO, PSO, and DE.

In terms of the solution quality obtained, FS acts as a measuring parameter and shows that the DE performs relatively better among all the considered EAs. However, DE is observed to consume the maximum amount of execution time in producing the outputs. This makes DE less preferable for larger datasets consisting of millions of nodes and links.

Convergence Measure (CM) denotes the number of function evaluations made by the algorithm in reaching the minimum objective function (Mof) fitness value. It can be inferred that the DE is able to achieve Mof comparatively faster, being the main reason for it to reach a high FS at the end of the entire run. DE also achieves a decent solution at

the initial iteration and also improves the solutions rapidly, allowing it to gain the highest SR. This appreciable performance of DE, comes with a downside in terms of SET, making it less scalable.

Table 8. CM scores of all EAs wrt. considered datasets

Name	Gene CM	ACO CM	PSO CM	DE CM
Dolphins	0.91%	100.00%	9.09%	0.91%
Human Protein	100.00%	36.36%	9.09%	15.45%
Wikipedia	100.00%	9.09%	9.09%	8.18%
Twitch	100.00%	9.09%	9.09%	20.91%
Cora	100.00%	18.18%	27.27%	16.36%
Twitter	100.00%	9.09%	27.27%	38.18%
Google plus	100.00%	9.09%	9.09%	0.91%
Epinions	100.00%	9.09%	9.09%	0.91%
Avg. CM	87.61%	25.00%	13.64%	12.73%

Overall, DE achieves the highest fitness scores, by still having the least number of fitness function evaluations and also in achieving desired fitness scores at a faster rate. This makes the Differential Evolution algorithm, the best among all the considered EAs in this experiment if the time consumed by the algorithm is not taken under serious consideration.

On the other hand, ACO stands achieves a great balance between fitness scores and evaluation time, in solving the IM problem. It achieves nearly 79% of the FS values of DE, but by consuming around 88% lesser time than DE. ACO is the fastest experimented algorithm among all the EAs, and the second-best after DE in terms of SR value. It can be preferred when a good solution is required in the shortest possible execution time, making it highly scalable for larger networks.

PSO was inferred to have the best CM value after DE, by making the second-least number of fitness evaluations to reach the Mof value. Genetic Algorithm was the second fastest-experimented algorithm followed after ACO. Although fast, GA was observed to fail to produce good quality results in terms of FS, SR, and CM.

5. Conclusion and Future Work

Influence Maximisation problem is one of the most researched areas in the field of social networks and has a number of practical applications. A desired number of nodes is selected from the entire dataset/network, to maximize the information spread. In the view of solving the IM problem, this paper evaluated various algorithms such as – Genetic Algorithm, Ant Colony Optimization, Particle Swarm Optimization, and Differential Evolution, on eight real-world social/complex network datasets. Considering the stochastic nature of EAs, the algorithms were compared using four performance metrics.

Upon assessment, it is concluded that the GE, PSO produces weaker quality of seed sets and also performs poorer in general when in comparison with other algorithms using the performance metrices. DE performs better in terms of solution quality and algorithmic efficiency but also consumes the maximum amount of time, relative to the other considered algorithms. On the other hand, In regard to balancing time requirements and solution quality, ACO does an excellent job in producing seed set for the provided networks. Being the fastest algorithm in terms of execution time, ACO is more scalable and can be preferred when the most influential nodes have to be found in the shortest possible time.

In future, we intend to hybridize the DE with ML algorithms, making it an Evolutionary Learning model to achieve better results and efficiency.

References

[1] C. L. Streeter and D. F. Gillespie, "Social network analysis," J. Social Service Res., vol. 16, nos. 1-2, pp. 201-222, 1993.

[2] Agash Uthayasuriyan, G. R. Ramya, and G. Jeyakumar. "Effective Link Prediction in Complex Networks Using Differential Evolution Based Extreme Gradient Boosting Algorithm." Advanced Network Technologies and Intelligent Computing: Second International Conference, ANTIC 2022, Varanasi, India, December 22–24, 2022, Proceedings, Part I. Cham: Springer Nature Switzerland, 2023. pp. 149-163.

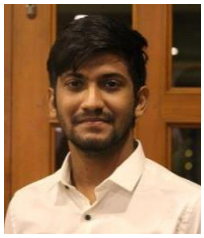
[3] A. H. L, S, A., G, A., P, S., and Dr. Sajeev G. P., “A Proximity Based Community Detection in Temporal Graphs”, in 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), 2020.

[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," 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, India, 2018, pp. 1290-1296.
[6] L. Wang, L. Ma, C. Wang, N. -G. Xie, J. M. Koh and K. H. Cheong, "Identifying Influential Spreaders in Social Networks Through Discrete Moth-Flame Optimization," in IEEE Transactions on Evolutionary Computation, vol. 25, no. 6, pp. 1091-1102, Dec. 2021.
[7] 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, Feb. 2018.
[8] L. Zhang, Y. Liu, F. Cheng, J. Qiu and X. Zhang, "A Local-Global Influence Indicator Based Constrained Evolutionary Algorithm for Budgeted Influence Maximization in Social Networks," in IEEE Transactions on Network Science and Engineering, vol. 8, no. 2, pp. 1557-1570.
[9] Weihua Li, Yuxuan Hu, Chenting Jiang, Shiqing Wu, Quan Bai, Edmund Lai, "ABEM: An adaptive agent-based evolutionary approach for influence maximization in dynamic social networks", Applied Soft Computing, Vol. 136, 110062, ISSN 1568 – 4946.
[10] C. Wang, J. Zhao, L. Li, L. Jiao, J. Liu and K. Wu, "A Multi-Transformation Evolutionary Framework for Influence Maximization in Social Networks," in IEEE Computational Intelligence Magazine, vol. 18, no. 1, pp. 52-67.
[11] D.-L. Nguyen, T.-H. Nguyen, T.-H. Do, and M. Yoo, "Probability- based multi-hop diffusion method for influence maximization in social networks," Wireless Pers. Commun., vol. 93, no. 4, pp. 903–916, Apr. 2017.
[12] A. Zareie, A. Sheikhahmadi, and M. Jalili, "Identification of influential users in social network using gray wolf optimization algorithm," Exp. Syst. Appl., vol. 142, Mar. 2020, Art. no. 112971.
[13] P. Surekha, P. Mohanaraajan, R. A., and Sumathi, S., "Ant Colony Optimization for Solving Combinatorial Fuzzy Job Shop Scheduling Problems", Int. Conf. on Communication and Computational Intelligence. Kongu Engineering College, Erode, India, 2010.
[14] Jayakumar V. and Raju, R., "A multi-objective genetic algorithm approach to the probabilistic manufacturing cell formation problem", South African Journal of Industrial Engineering, vol. 22, no. 1, pp. 199-212, 2011.
[15] 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.
[16] 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.
[17] 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.
[18] 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.
[19] S. Mirjalili, ''Genetic algorithm,'' in Evolutionary Algorithms and Neural Networks. Cham, Switzerland: Springer, 2019, pp. 43–55.
[20] Dhanalakshmy, D.M., Akhila, M.S., Vidhya, C.R., Jeyakumar, G.: Improving the search efficiency of differential evolution algorithm by population diversity analysis and adaptation of mutation step sizes. Int. J. Adv. Intell. Paradig. 15(2), 119–145 (2020).
[21] Simpson, A. R. Maier, H. R., Foong, W. K., Phang, K. Y., Seah, H. Y., and Tan, C. L., 2001, 'Selection of parameters for ant colony optimization applied to the optimal design of water distribution systems', in Proc., Int. Congress on Modeling and Simulation. Canberra, Australia, pp. 1931– 1936.
[22] Alan Piszcz, Terence Soule, (2006) "Genetic Programming: Analysis of Optimal Mutation Rates in a Problem with Varying Difficulty", in the proceedings of Artificial Intelligence Research Society Conference, vol.19. Flairs, Florida, pp.451-456,2006.
[23] Kunegis, J.: KONECT—the Koblenz network collection. In: Proceedings of the 22nd International Conference on World Wide Web, pp. 1343–1350 (2013).
[24] Z. Liang, Q. He, H. Du and W. Xu, "Targeted influence maximization in competitive social networks", *Information Sciences*, vol. 619, pp. 390-405, 2023.
[25] Zareie, A., Sakellariou, R. Influence maximization in social networks: a survey of behaviour-aware methods. Soc. Netw. Anal. Min. 13, 78 (2023).

Authors' Profiles

Agash Uthayasuriyan is a final-year B. Tech undergraduate student in Electrical and Computer Engineering at Amrita Vishwa Vidyapeetham, Coimbatore, India. His area of interest includes Evolutionary Computation, Machine Learning, and in Cloud computing.

He has a particular interest in exploring hybrid models that uses the best of various algorithms. His published research in the Evolutionary Learning domain involving the Extreme gradient boost algorithm and Differential Evolution has proven to yield better results in predicting the links present in social networks. His knowledge and understanding in the field of Artificial Intelligence have made him contribute to several reputable conferences and journals.



Hema Chandran G, a final year B. Tech Electrical and computer engineering student at Amrita Vishwa Vidyapeetham in Tamil Nadu, India.

He is passionate about the field of artificial intelligence, data analytics, social networks, and graph neural networks. With his deep knowledge of these domains, he has been actively involved in research projects and has contributed to several publications.

Kavvin UV is a final-year undergraduate student, pursuing Electrical and Computer Engineering at Amrita Vishwa Vidyapeetham, Coimbatore, India.

Being interested in Machine Learning, he has a specific passion towards Reinforcement Learning. He has published publications in conferences and journals as a result of his scientific work.

Sabbineni Hema Mahitha is currently pursuing her B. Tech degree in Electrical and Computer Engineering at Amrita Vishwa Vidyapeetham, Coimbatore, India. Her area of interest ranges from using mathematical models to Machine learning models for solving complex problems.

She has worked on research articles for various conferences and journals, and her area of interest is using data-driven models for business analysis and decision making.

Dr. G. Jeyakumar received his B.Sc. degree in Mathematics in 1994, M.C.A degree (under the faculty of Engineering) in 1998 from Bharathidasan University, and a Ph.D. degree in Distributed Differential Evolution Algorithm in 2013, from Amrita Vishwa Vidyapeetham University, Tamil Nadu, India. He is currently a Professor in the Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham University, Tamil Nadu, India since 2000.

His research interests include Parallelization and Applications of Evolutionary Algorithms, Artificial Intelligence Techniques, and Human Modeling. He has published numerous papers in reputable journals and conference proceedings, out of which the majority of the publications are indexed in SCOPUS. He has got the best paper awards for a few of his publications. He has guided many students' projects/theses belonging to the courses B. Tech, M. Tech (CSE), M. Tech (Automotive Eng.), M.C.A, and M. Phil and currently guiding Ph.D. scholars, many UG and PG students.

How to cite this paper: Agash Uthayasuriyan, Hema Chandran G, Kavvin UV, Sabbineni Hema Mahitha, Jeyakumar G, "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. DOI:10.5815/ijmecs.2024.02.07