Chapter 3

Influence Maximization using Learning-automata based Particle Swarm Optimization

This chapter studies a learning-based evolutionary technique and utilized for efficient identification of influential users by optimizing influence evaluation function.

3.1 Introduction

A fundamental problem in influence maximization problem is inapplicability of greedy method for complex diffusion models. There are some diffusion models like the competitive threshold model [42] may not be sub-modular and monotonic, which leads to the inapplicability of

greedy-based methods. Therefore, we need to address the IM problem for a wide range of diffusion models. This issue is address by either breaking the boundary of submodularity or developing and adopting more general techniques like heuristics, meta-heuristics, and nature-inspired optimization methods. Nature-inspired techniques can be applied to any network, regardless of whether it possesses a community structure or not. Simply requires to define appropriate objective function that is to optimized. Discrete particle swarm optimization (DPSO) [44] is one of the most popular method for optimizing NP-hard problems due to its simplicity and efficiency. The major weakness of DPSO is its premature convergence which leads to trapping in local optima. To maintain a trade-off between local and global search process, Hasanzadeh et al. [45] proposed a learning automata (LA) based DPSO named as DPSOLA.

In this chapter, DPSOLA [45] is studied and utilized for efficient identification of seed users by optimizing local objective function. A learning-automata based discrete particle swarm optimization (LAPSO-IM) is designed to identify seed users. The major contributions of the chapter are as follow.

- Exploring the first objective, the local influence evaluation function expected diffusion value (EDV) [46] is extended to approximate within the two-hop area as $EDV^{(2)}(S)$.
- Further exploring the second objective, a learning-automata based PSO approach is utilized to optimize local influence evaluation

function by redefining the velocity and position rules based on learning automata.

- The LAPSO-IM algorithm is designed and experimentally compared with the state-of-the-art algorithms in terms of quality and efficiency under traditional diffusion models.
- The empirical results on six real-world networks show that the proposed algorithm LAPSO-IM performs better than the base method DPSO in terms of influence spread with almost same running time. LAPSO-IM is more time-efficient than base method IMLA with the same level of influence spread.

3.1.1 Discrete Particle Swarm Optimization

Kennedy and Eberhart [44] were first to introduced a population-based optimization meta-heuristic, viz. PSO. PSO is inspired by the behaviors of fish schools and bird flocks. In PSO, a population (swarm) pop of individuals (particles), moves inside a bounded d-dimensional search space and cooperates with each-other to find best possible solution for a given problem. Each particle $j \in \{1,2,3,\ldots,pop\}$ has two vectors, position $pos(j)$ and velocity $V(j)$. Let $pos(j) = (p_{j1}, p_{j2}, p_{j3}, \ldots, p_{jd})$ and $V(j) = (v_{j1}, v_{j2}, v_{j3}, \ldots, v_{jd})$ represents position and velocity vector of particle j respectively.

Originally, PSO was introduced to solve global continuous optimization problems. Later, Kennedy and Eberhart [151] proposed PSO for discrete optimization problems. Recently, a lot of work is done in PSO and many versions of PSO have been introduced. The version proposed by Shi and Eberhart [152], is used in most of the applications. The update rules for position and velocity are defined by Shi and Eberhart [153], as follows.

$$V(j) \leftarrow \delta V(j) + c_1 r_1 (pbest(j) - pos(j)) + c_2 r_2 (gbest - pos(j)) \quad (3.1)$$

$$pos(j) \leftarrow pos(j) + V(j) \quad (3.2)$$

where δ represents inertia weight, c_1 and c_2 are learn factors, $r_1 \in [0,1]$ and $r_2 \in [0,1]$ are random numbers, $pbest(j)$ is personal best position of particle j, and $gbest$ represents global best position in population.

3.1.2 Learning Automata

Learning Automata (LA) is a decision-making machine with learning capability [154]. LA performs a finite set of actions α. Each action α_i is evaluated by a probability p_i and evaluation result generates a feedback β. The objective of LA is learn to find the optimal solution. LA is formulated as quadruple $\langle \alpha, \beta, p, T \rangle$, where $\alpha = \{\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_r\}$, $\beta = \{\beta_1, \beta_2, \beta_3, \ldots, \beta_s\}$, $p = \{p_1, p_2, p_3, \ldots, p_r\}$, and T represent an action set, response set, probability set, and the reinforcement process respectively. A linear reinforcement process [155] updates probability

vector p for action $\alpha_i \in \alpha$ with $\beta \in \{0,1\}$ as follows.

$$p_i = \begin{cases} p_i + a_r[1-p_i] & \beta = 0 \\ (1-b_p)p_i & \beta = 1 \end{cases} \tag{3.3}$$

$$p_j = \begin{cases} (1-a_r)p_j & \beta = 0 \\ \frac{b_p}{r-1} + (1-b_p)p_j & \beta = 1 \end{cases} \tag{3.4}$$

where $a_r \in (0,1)$ and $b_p \in (0,1)$ represent reward and penalty parameters respectively. FIGURE 3.1 shows the learning process of learning automata.

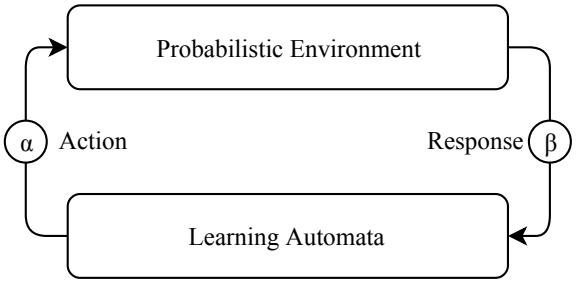


FIGURE 3.1: An Automaton Learning in LA using a Probabilistic Environment

3.2 Proposed Approach

In this section, the proposed algorithm LAPSO-IM for maximizing influence spread in social networks is presented. To maintain a trade-off between local and global search, LAPSO-IM uses LA to configure particles behavior in PSO. Firstly, the proposed framework is discussed then detail of each component is described.

Learning-automata based PSO [45] has four phases: initialization, update, learning and final.

- **Initialization:** In this phase, an objective function is defined. Parameters values used in PSO and LA are assigned. The Initial velocity of each particle is assigned with zero. The position and personal best (pbest) of particles are assigned based on node degree centrality. Also, the global best (solution) is assigned using their calculated fitness values.
- **Update:** In this phase, iteration is performed based on the termination condition. All the particles in swarm move to a new position using their velocity. Each iteration is divided into sub-steps like, selection of LA action, velocity evaluation, fitness value estimation, finding *pbest* and *gbest* vectors.
- **Learning:** In this phase, the LA trains system based on their previous responses. It updates the probability vector based on learning machine response.
- **Final:** This phase performs constraints handling. To handle the boundary constraints, many methods exist.

The principal components of the proposed algorithm LAPSO-IM are outlined as follow.

- **Swarm.** The swarm refers to the population *pop* of individuals (particles).

- **Particle.** The particles are agents of LAPSO-IM to perform optimization. The particles move towards the current best solution based on LA action α_i.

- **Fitness function.** In order to get the accurate influence spread of a seed set, the existing algorithms always need to run at least ten thousand time which is very time-consuming. As a result, Jiang et al. [46] proposed *expected diffusion value* (EDV) under IC model. In literature, there are several other objective functions of IM are presented such as MaxDegree [11], Degree Discount [24], Diffusion Degree [26], CUCB [156], and DILinUCB [156], etc. The objective functions MaxDegree, Degree Discount, and Diffusion Degree only consider node degree, and avoid their influence probabilities to their direct and indirect neighbors unlike EDV. Therefore, these objective functions generate poor quality seed than EDV with almost the same efficiency. CUCB and DILinUCB are based on pairwise influence feedback model, and generate seed nodes with approximate influence spread with low efficiency than EDV. Hence, it is essential to design a function that acquires a good trade-off between influence spread and running time to compute the influence spread of a node set. Therefore, algorithm selects EDV as fitness function of LAPSO-IM.

EDV considers influence in the one-hop area (direct neighbors). The expected diffusion value of seed set S in the one-hop area is computed as follows.

$$EDV(S) = \sigma_0(S) + \sigma_1(S) = \quad k + \sum_{u \in N(S) \setminus S} (1 - \prod_{(u,v) \in E, v \in S} (1 - p_{uv})) \tag{3.5}$$

where p_{uv} represents influence probability of u to v. In social networks, the influence spread of a user opinion is limited to within its local region like three-hop area [157], two-hop area [158]. Following these studies, algorithm utilize an approximate influence estimation function $EDV^{(2)}(S)$ within the two-hop area as fitness function. Expected diffusion value in two-hop area $EDV^{(2)}(S)$ can be estimated as follows.

$$EDV^{(2)}(S) = \sum_{i=0}^{i=2} \sigma_i(S) \tag{3.6}$$

$$EDV^{(2)}(S) = k + \sum_{u \in S, v \in N_{out}^{NA}(u)} p_{u,v} + \sum_{v \in \{N_{out}^{A}(S) \setminus S\}, w \in N_{out}^{NA}(v)} p_A(v) . p_{v,w} \tag{3.7}$$

where $N_{out}^{NA}(u)$ represents in-active out-neighbors of u, and $N_{out}^{A}(S)$ represents active out-neighbors of S. The fitness function of LAPSO-IM is given by local influence spread function $EDV^{(2)}(S)$ depicted in Equation 3.7.

- **Learning automata.** Each LA performs one of three actions: *global*

search, *local search*, and *global and local combined search* [45]. Based on action α_i, the reinforcement signal β is generated. The value of response/feedback by the probabilistic environment is calculated by Equation 3.8.

$$\beta = \begin{cases} 0 & f(pos(j)) > f(pos_{prev}(j)) \\ 1 & \text{otherwise} \end{cases} \tag{3.8}$$

- **Velocity.** The velocity of a particle j is represented by vector $V(j) = (v_{j1}, v_{j2}, v_{j3}, \ldots, v_{jk})$ where $k = |S|$. Suppose that LA selects action *local search* (LS) then particle moves toward the best position with $\delta = 0$ and velocity $V(j)$ of particle j is calculated as follows.

$$\begin{aligned} V(j) \leftarrow F(r_1(a + \frac{1}{R_{it}})(pbest(j) \cap pos(j)) \\ +(b - \frac{1}{R_{it}})(pbest(j) \cap pos(j))) \end{aligned} \tag{3.9}$$

where $a, b \in [0,1]$ are the weight index (learn factor), $r_1 \in [0.6, 1.2]$ represents random number, and $R_{it} = (I_{max} + 1) - I$. I_{max} and I denote maximum number of iteration and current iteration number respectively. If LA selects action *global search* (GS) then velocity $V(j)$ is calculated as follows.

$$V(j) \leftarrow F(r_1(a + \frac{1}{R_{it}})(pbest(j) \cap pos(j)) + cr_2(gbest \cap pos(j))) \tag{3.10}$$

where c denotes acceleration constant, and $r_2 \in [0,1]$ represents

random number. If LA selects action *global and local search combined* (GLC) then velocity $V(j)$ is calculated as follows.

$$V(j) \leftarrow F(\delta V(j) + r_1(a + \frac{1}{R_{it}})(pbest(j) \cap pos(j)) + (b - \frac{1}{R_{it}}) \\ (pbest(j) \cap pos(j)) + cr_2(gbest \cap pos(j))) \tag{3.11}$$

where $\delta \in [0,1]$ is inertia weight. Parameter δ is used to control degree of local and global search. The operator $\cap$ used in same sense as intersection operator. For example, suppose two vectors X and Y are {A,D,F} and {A,B,D} respectively, then $X \cap Y = \{0,1,0\}$. The argument of function $F(.)$ is a position vector. Suppose that argument of $F(.)$ is $pos(i) \in \{pos(1), pos(2), \ldots, pos(pop)\}$ then function $F(pos(i))$ is denoted as $\{f_1(p_{i1}), f_2(p_{i2}), \ldots, f_k(p_{ik})\}$. The value of $f_l(p_{il}) \in F(pos(i))$ is computed as follows.

$$f_l(p_{il}) = \begin{cases} 0 & p_{il} < 1 \\ 1 & \text{otherwise} \end{cases} \tag{3.12}$$

For example, the argument of $F(.)$ is (1.2,0.78,2.3) then $F(1.2, 0.78, 2.3) = (1,0,1)$.

- **Position.** The position of a particle j is represented by vector $pos(j) = (p_{j1}, p_{j2}, p_{j3}, \ldots, p_{jk})$ where $k = |S|$. Each particle's position give a solution. In each iteration, position of particle j is

estimated as follows.

$$pos(j) = pos(j) \otimes V(j) \tag{3.13}$$

where the operator $\otimes$ is defined as $pos(j) \otimes V(j) = pos'(j) = \{p'_{j1}, p'_{j2}, \ldots, p'_{jk}\}$ and $p'_{jl} \in pos'(j)$ is computed as follows.

$$p'_{jl} = \begin{cases} p_{jl} & v_{jl} = 0 \\ replace(p_{jl}, N) & v_{jl} = 1 \end{cases} \tag{3.14}$$

where $N \in G$. For example, $pos(j)$ and $V(j)$ are {B,F,G} and {0,1,0} respectively, then $pos(j) = (B, D, G)$.

- **pbest(j).** The personal best position of particle j is defined as $pbest(j) = \{pbest_{j1}, \ldots, pbest_{jk}\}$.
- **gbest.** The global best position of swarm is defined as $gbest = \{gbest_1, \ldots, gbest_k\}$ which gives best possible solution of IM problem.

3.3 Algorithm

The main **Algorithm 2** takes two inputs, an influence graph G and the size of seed set k. LAPSO-IM performs the local and global search based on LA response in influence graph G using PSO and finds k most influential users. FIGURE 3.2 presents the flow chart for working of the algorithm.

Algorithm 2: LAPSO-IM(G, k): The Proposed Algorithm

Input: Influence graph G, seed size k
Output: Seed set S

$j \leftarrow 1$ ▷ Initialization phase
Assign values to the parameters of PSO and LA
for $j \leq pop$ **do**
  $pos(j) \leftarrow$ Select k highest out-degree nodes
  $pbest(j) \leftarrow$ Select k highest out-degree nodes
  $l \leftarrow 1$
  for $l \leq k$ **do**
    if $rand(0,1) < 0.5$ **then**
      $p_{jl} \leftarrow replace(p_{jl}, V \setminus pos(j))$
      $pbest_{jl} \leftarrow replace(pbest_{jl}, V \setminus pbest(j))$
    $V_{jl} \leftarrow 0$
    $l \leftarrow l + 1$
  Estimate particle fitness value $f(pos(j))$
  $j \leftarrow j + 1$
$gbest \leftarrow$ Select best fitness value particle $pos(i)$
$I \leftarrow 1$ ▷ Update phase
$p \leftarrow \{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$
while $I \leq I_{max}$ **do**
  $j \leftarrow 1$
  for $j \leq pop$ **do**
    LA select an action α_i using probability vector p
    if $\alpha_i = local\ search$ **then**
      Update velocity vector $V(j)$ using Equation 3.9
    if $\alpha_i = global\ search$ **then**
      Update velocity vector $V(j)$ using Equation 3.10
    if $\alpha_i = global\ and\ local\ combined\ search$ **then**
      Update velocity vector $V(j)$ using Equation 3.11
    $pos_{prev}(j) \leftarrow pos(j)$
    Update position vector $pos(j)$ using Equation 3.13
    Update particle's fitness $f(pos(j))$
    $pbest(j) \leftarrow$ LocalSearch($pbest(j), pos(j)$) ▷ See **Algorithm 3**
    $\beta \leftarrow$ Generate LA response β using Equation 3.8 ▷ Learning Phase
    Update probability vector p using Equation 3.3 & 3.4
    $j \leftarrow j + 1$
  $gbest' \leftarrow$ Select best fitness value particle $f(pos(i))$
  $gbest \leftarrow max(gbest, gbest')$
  $I \leftarrow I + 1$
$S \leftarrow gbest$
Return S

Algorithm 3: LocalSearch*(pX,X)*

Input: Personal best vector pX, position vector X
Output: Y

```
1  Y ← pX
2  Temp ← {pX \ (pX ∩ X)}
3  foreach Y_j ∈ Temp do
4  |   Y_j ← replace(Y_j, N(Y_j))
5  |   if f(Y) > f(pX) then
6  |   |   pX ← Y
7  |   else
8  |   |   Y ← pX
9  Y ← pX
10 Return Y
```

First of all, line 1 assigns j to 1. Line 2 assigns values of required parameters of PSO and LA likes population size pop, the maximum number of iteration I_{max}, inertia weight δ, weight indexes (a,b), reward parameter a_r, and penalty parameter b_p etc. The **for** loop in lines 3-14, initializes position, velocity, and *pbest* vectors of particle j. Lines 4-5 assign position *pos* and personal best *pbest* vector with k highest out-degree nodes. Line 6 initializes l with 1. The **for** loop in lines 7-12, promotes diversity in positions of particles. Line 13 computes fitness of particle j. Line 14 performs increment in l. Line 15 finds solution vector *gbest*. Line 16 initializes iteration number I with 1. Line 17 initializes the probability vector p of learning automata. The **while** loop in lines 18-37, finds solution *gbest* iteratively. Line 19 initializes j with 1. The **for** loop in lines 20-34, updates position and velocity vector of particles repeatedly. Line 21 selects an action $\alpha_i \in \alpha$ based on probability vector p. Lines 22-23 check, if selected action is *local search* then velocity is estimated

using Equation 3.9. Lines 24-25 check, if selected action is *global search* then velocity is estimated using Equation 3.10. Lines 26-27 check, if selected action is *global and local combined search* then velocity is estimated using Equation 3.11. Line 28 stores previous position vector $pos(j)$ of particle j before updating the position vector. Line 29 updates position vector $pos(j)$ of particle j. Lines 30 calculates particle fitness. Line 31 updates local best position vector $pbest(j)$ of particle j. Line 32 calculates the reinforcement (response) signal β using Equation 3.8. Line 33 updates probability vector p of learning automata. Line 34 performs the increment operation on j. Line 35 estimates *gbest* of the current iteration. Line 36 finds global best position vector *gbest* until current iteration. Line 37 performs the increment operation on I. LAPSO-IM repeats the iterations of the **while** until the termination condition $I \leq I_{max}$ is satisfied. Line 38 assigns *gbest* to seed set S. Lastly, line 39 returns the seed set as the solution of given problem.

Algorithm 3 takes two position vectors pX and X as input. It performs local search and finds best position vector $pbest(j)$ for each particle j. First of all, line 1 stores position vector pX into Y. Line 2 selects nodes from vector pX which are not present in vector X. The **foreach** loop in lines 3-8, finds personal/local best position vector. Line 4 replaces Y_j with its neighbors. Lines 4-8 update local position vector. Line 9 assigns vector pX to Y. Lastly, Line 10 returns local best position vector Y.

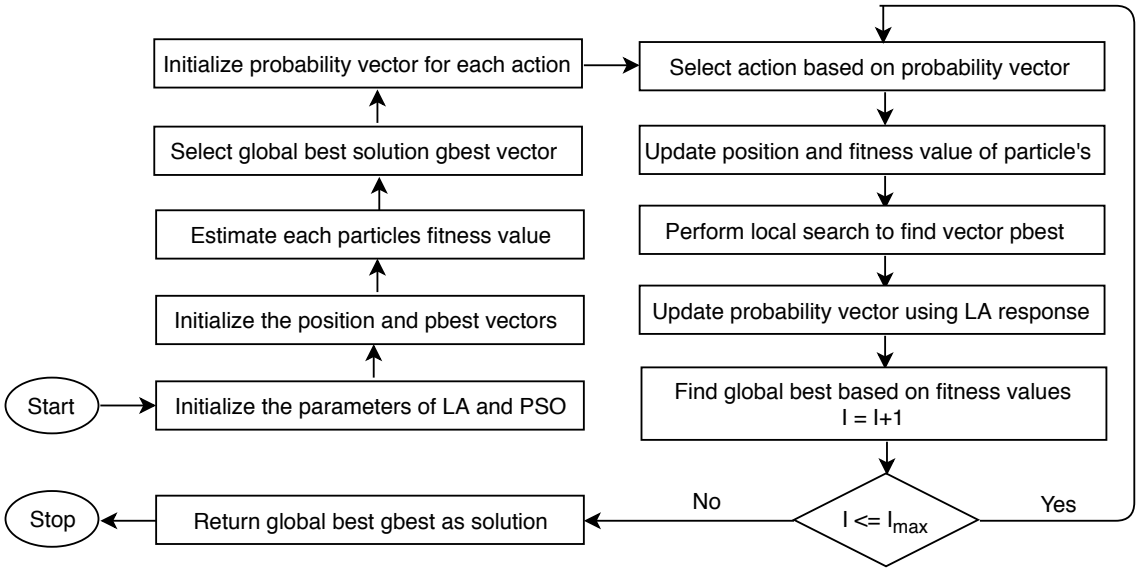


FIGURE 3.2: The Work Flow Diagram of the Proposed Algorithm LAPSO-IM

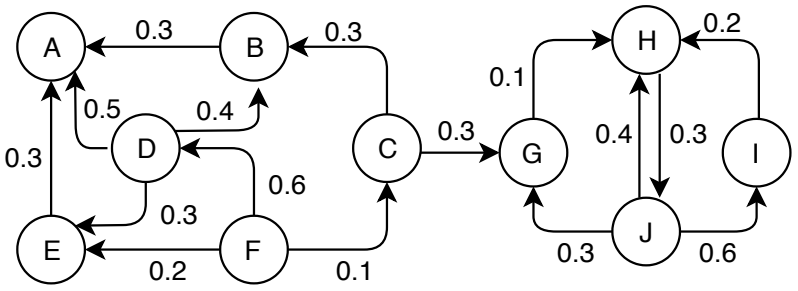


FIGURE 3.3: A Running Example Graph

3.3.1 Applying the Algorithm

To explain the working of the proposed algorithm, we take an example graph as shown in FIGURE 3.3. Here, we consider LAPSO-IM system as a complete undirected graph $G(V,E)$ where V represents particles in swarm and E represents edges between them. Each particle $P(j)$ connects with other particles i.e, $|E| = \frac{|V|(|V|-1)}{2}$. Suppose that population of swarm $pop = 5$.

Initialization: Suppose, $pop \leftarrow 5$, $I_{max} \leftarrow 3$, $\delta \leftarrow 0.6$, $a \leftarrow 0.8$, $b \leftarrow 0.6$, $c \leftarrow 0.5$, $a_r \leftarrow 0.8$, $b_p \leftarrow 0.6$, $k \leftarrow 2$, $\beta \leftarrow \{0,1\}$, and diffusion model $M \leftarrow WC$. Initially, velocity $V(j)$ of each particle P_j is assigned to zero. Position $pos(j)$ and personal best $pbest(j)$ are initialized by k highest degree nodes. To promote swarm diversity, LAPSO-IM uses replace method (lines 7-12). Let us assume that particle P_1 initializes $pos(1)$ and $pbest(1)$ with (B,E) and (D,E) respectively. Fitness of particle P_1 and P_2 are calculated as $f(1) = 2 + (0.3 \oplus 0.3) + (0+0) = 2.3$ and $f(2) = 2 + ((0.5+0.4+0.3) +0.3) + (0+0+0+0.3*0.1) = 3.53$ respectively. Similarly, we estimate fitness of particle P_3, P_4, and P_5 as shown in TABLE 3.1. Probability vector is initialize with $p(LS,GS,CLC) = (1/r, 1/r, 1/r) = (0.33, 0.33, 0.33)$.

Iteration: Iteration phase starts at $I = 1$. Each particle P_i updates position and velocity vectors as follows.

- **Particle** P_1. Suppose LA chooses an action $\alpha_i = LS$ based on probability vector p then velocity $V(1)$ is calculated using Equation 3.9. So $V(1)$ is estimated as $V(1) = F(r_1(a + \frac{1}{R_{it}})(pbest(1) \cap pos(1)) + (b - \frac{1}{R_{it}})(pbest(1) \cap pos(1))) = F(0.4(0.8 + \frac{1}{3+1-1})(DE \cap BE) + (0.6 - \frac{1}{3+1-1})(DE \cap BE)) = F(0.452(1,0) + 0.27(1,0)) = (0,0)$. Using Equation 3.13, position of P_1 is estimated as $pos(1) = pos(1) \otimes V(1) = (B,E) \otimes (0,0) = (B,E)$. $f(1) = 2.3$, $pbest = (D,E)$. Now we estimate response β using Equation 3.8, $f(1) \leq f_{prev}(1)$, $\beta = 1$. Probability vector p is updated using Equations 3.3-3.4. $p = ((1-b_p)p_1, \frac{b_p}{r-1} + (1-b_p)p_1, \frac{b_p}{r-1} +$

$(1 - b_p)p_1) = ((1 - 0.6) * 0.33, 0.3 + (1 - 0.6) * 0.33, 0.3 + (1 - 0.6) * 0.33) = (0.13, 0.43, 0.43)$.

- **Particle** P_2. LA chooses action $\alpha_i = GLC$ using vector p. Using Equation 3.11, velocity is computed as $V(2) = F(0.6(0,0) + 0.7(0.8 + 0.33)(EG \cap DC) + (0.6 - 0.33)(EG \cap DC) + 0.5 * 0.2(DH \cap DC)) = 0.79(1,1) + 0.27(1,1) + 0.1(0,1) = (1,1)$. Position of P_2 is $pos(2) = (D,C) \otimes (1,1) = (D,J)$. Fitness value of P_2 is estimated as $f(2) = 2 + (0.5 + 0.4 + 0.3) + (0.6 + 0.4 + 0.3) = 4.5$. $pbest = (E,H)$ and $\beta = 0$. Probability vector is calculated as $p = ((1 - 0.8) * 0.13, (1 - 0.8) * 0.43, 0.43 + 0.8 * (1 - 0.43)) = (0.02, 0.08, 0.80)$.

- **Particle** P_3. LA selects action $\alpha_i = GLC$ using vector p. Using Equation 3.11, velocity is computed as $V(3) = F(0.6(0,0) + 0.6(0.8 + 0.33)(DB \cap DH) + (0.6 - 0.33)(DB \cap DH) + 0.5 * 0.5(DH \cap DH)) = 0.67(0,1) + 0.27(0,1) + 0.25(0,0) = (0,0)$. Position is $pos(3) = (D,H) \otimes (0,0) = (D,H)$. $f(2) = 3.77$, $pbset(3) = (D,G)$, and $\beta = 1$. p is calculated as $(0.3 + (1 - 0.6) * 0.02, 0.3 + 0.4 * 0.08, (1 - 0.6) * 0.8) = (0.30, 0.33, 0.32)$.

- **Particle** P_4. LA chooses action $\alpha_i = GS$ using vector p. Using Equation 3.10, velocity is computed as $V(4) = F(0.4(0.8 + 0.33)(FJ \cap FI) + 0.5 * 0.6(DH \cap FI))$

$= 0.45(0,1) + 0.3(0,0) = (0,0)$. Position is $pos(4) = (F,I) \otimes (0,0)$ $= (F,I)$. $f(4) = 3.91$, $pbset(4) = (F,J)$, and $\beta = 1$. p is calculated as $(0.3 + (1 - 0.6) * 0.3, (1 - 0.6) * 0.33, 0.3 + 0.4 * 0.32)$ $= (0.42, 0.13, 0.43)$.

- **Particle** P_5. LA selects action $\alpha_i = GLC$ based on vector p. Using Equation 3.11, velocity is computed as $V(5) = F(0.6(0,0) + 0.3(0.8 + 0.33)(HF \cap DA) + (0.6 - 0.33)(HF \cap DA) + 0.5 * 0.4(DH \cap DA)) = 0.33(1,1) + 0.27(1,1) + 0.2(0,1) = (0,0)$. Position is $pos(5) = (D,A) \otimes (0,0) = (D,A)$. $f(5) = 3.20$, $pbset(5) = (D,J)$, and $\beta = 1$. p is calculated as $(0.3 + (1 - 0.6) * 0.42, 0.3 + 0.4 * 0.013, (1 - 0.6) * 0.43) = (0.46, 0.35, 0.172)$.

The process continues until $I \leq I_{max}$ in the same manner. TABLE 3.1 gives the detailed description of each iteration.

Final: Finally the algorithm returns the desired output $gbest = \{D,J\}$ as seed set.

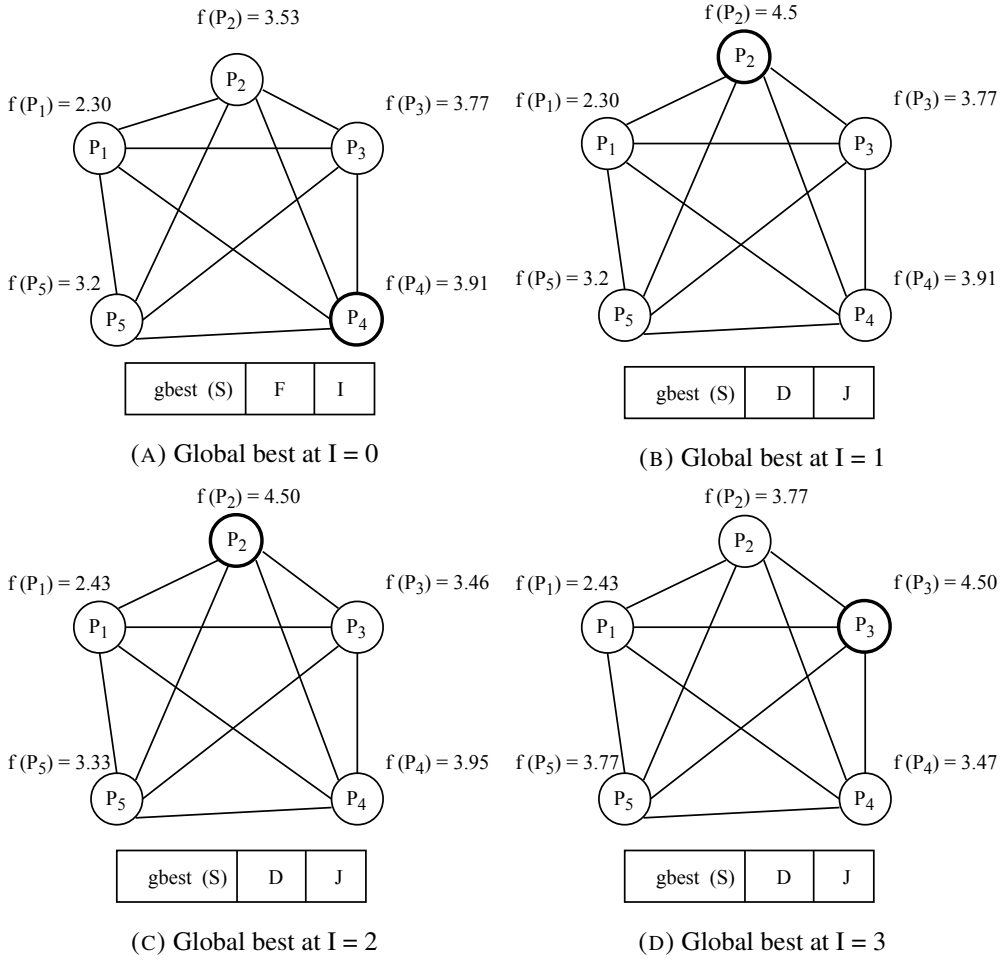
FIGURE 3.4 shows working of LAPSO-IM system. FIGURE 3.4a shows the fitness of each particle at iteration $I = 0$. Particle P_4 has highest fitness value so position of P_4 is considered as current global best $gbest$. FIGURE 3.4b shows that particle P_2 has highest fitness value at iteration $I = 1$, so that $gbest = \{D,J\}$. Similarly, FIGURE 3.4c and 3.4d show the $gbest$ at $I = 2$ and $I = 3$ respectively. The final global best $gbest$ is considered as seed set (solution).

TABLE 3.1: The Working of Update Phase of LAPSO-IM in Running Example

Input: $pop \leftarrow 5$, $I_{max} \leftarrow 3$, $\delta \leftarrow 0.6$, $a \leftarrow 0.8$, $b \leftarrow 0.6$, $c \leftarrow 0.5$, $a_r \leftarrow 0.8$, $b_p \leftarrow 0.6$, $k \leftarrow 2$											
I	j	$V(j)$	$pos(j)$	$f(j)$	$pbest(j)$	$p(LS,GS,GLC)$	β	α_i	r_1	r_2	$gbest$
0	1	(0,0)	(B,E)	2.30	(D,E)	$(0.33,0.33,0.33)$	–	–	–	–	(F,I)
	2	(0,0)	(D,C)	3.53	(E,G)	(0.33,0.33,0.33)	–	–	–	–	
	3	(0,0)	(D,H)	3.77	(D,B)	(0.33,0.33,0.33)	–	–	–	–	
	4	(0,0)	(F,I)	3.91	(F,J)	(0.33,0.33,0.33)	–	–	–	–	
	5	(0,0)	(D,A)	3.20	(H,F)	(0.33,0.33,0.33)	–	–	–	–	
1	1	(0,0)	(B,E)	2.30	(D,E)	(0.13,0.43,0.43)	1	LS	0.4	0.8	(D,J)
	2	(1,1)	(D,J)	4.50	(E,H)	(0.02,0.08,0.80)	0	GLC	0.7	0.2	
	3	(0,0)	(D,H)	3.77	(D,G)	(0.30,0.33,0.32)	1	GLC	0.6	0.5	
	4	(0,0)	(F,I)	3.91	(F,J)	(0.42,0.13,0.43)	1	GS	0.4	0.6	
	5	(0,0)	(D,A)	3.20	(D,J)	(0.46,0.35,0.17)	1	GLC	0.3	0.4	
2	1	(1,0)	(G,E)	2.43	(D,E)	(0.89,0.07,0.03)	0	LS	0.8	0.6	(D,J)
	2	(0,0)	(D,J)	4.50	(D,H)	(0.35,0.32,0.31)	1	LS	0.6	0.4	
	3	(0,1)	(D,I)	3.46	(D,H)	(0.14,0.43,0.42)	1	LS	0.7	0.4	
	4	(0,1)	(F,C)	3.95	(F,J)	(0.03,0.80,0.08)	0	GS	0.8	0.5	
	5	(0,1)	(D,G)	3.33	(D,J)	(0.01,0.96,0.02)	0	GS	0.6	0.5	
3	1	(0,0)	(G,E)	2.43	(D,E)	(0.30,0.38,0.31)	1	GS	0.4	0.8	(D,J)
	2	(0,1)	(D,H)	3.77	(D,J)	(0.42,0.15,0.43)	1	GS	0.9	0.6	
	3	(0,1)	(DJ)	4.50	(DH)	(0.08,0.03,0.80)	0	GLC	0.7	0.5	
	4	(0,1)	(F,A)	3.47	(F,J)	(0.31,0.30,0.32)	1	GLC	0.3	0.6	
	5	(0,1)	(D,H)	3.77	(D,J)	(0.06,0.06,0.86)	0	GLC	0.2	0.7	
Output: Seed set $\leftarrow \{D,J\}$, Active set $\leftarrow \{A,B,D,E,G,H,I,J\}$, Influence spread $\leftarrow 4.50$											

3.3.2 Complexity Analysis

In this section, we analyze the time complexity of the proposed algorithm. For each particle, the initialization phase (lines 1-17) needs $O(k|V|)$, $O(k|V|)$, and $O(k.D_{avg}(N)^2)$ time to initializes position, *pbest*, and fitness respectively, where $D_{avg}(N)$ is the average degree of the network. To initialize the *gbest*, it takes $O(n)$ time, where $|pop| = n$. Hence, the time complexity of initialization phase is $O(n(k|V| + k|V| + k.D_{avg}(N)^2) + n)$. In update phase, velocity of each particle is estimated in $O(k.\log k)$ time (lines 21-27). The position of a particle is calculated in $O(k)$ time (line 29). The fitness of each particle is calculated in $O(kD_{avg}(N)^2)$ time (line 30). The personal best *pbest* is updated in time $O(k.D_{avg}(N))$ (line 31).



(A) Global best at I = 0

(B) Global best at I = 1

(C) Global best at I = 2

(D) Global best at I = 3

FIGURE 3.4: The Working Example of LAPSO-IM when $pop = 5$

The probability vector is updated in $O(r)$ time (line 33). The global best *gbest* is updated in $O(k)$ time (lines 36-37). The overall time complexity of update phase is $O(I_{max}(n(k.\log k + k + k.D_{avg}(N)^2 + k.D_{avg}(N) + r) + k))$. Therefore, the worst case time complexity of LAPSO-IM is $O(I_{max}.n.k.(\log k + D_{avg}(N)^2) + n.k.|V|)$.

3.4 Empirical Analysis

In this section, firstly, the experimental setup information is provided along with the dataset, seeding strategies, and probability distribution for diffusion models. Further, discussion of the performance analysis regarding influence spread and efficiency along with parameter analysis are provided. Finally, section illustrate the statistical test to show the significant difference between the proposed method and the compared seeding methods.

3.4.1 Experimental Setup

All the experiments performed on six real-world network datasets: NetInfective [139], NetScience [142], Ca-GrQc [141], NetHEPT [24], Gnutella30 [143], and NetPHY [144]. The performance of proposed algorithm is tested against five seeding strategies: Degree Discount [24], Diffusion Degree [26], CELF++ [29], IMLA [48], and DPSO [47]. The algorithm utilize well-known probability distributions to assign influence probabilities. The propagation probability for each edge follows a uniform distribution. The propagation probability in IC models assigned based on uniform distribution over $\{0.1,0.01,0.001\}$. For the remaining two models (WC, LT), propagation probability $p_{x,y}$ is assigned by $\frac{1}{indegree(y)}$. The activation probability of a node x follows uniform distribution over [0,1]. Each of the methods is implemented using C++

language and executed on DEV-C++. All of the algorithms run 20 times independently on a 64-bit window's PC with Intel(R) Core(TM) i7-7700 CPU@ 3.60GHz processor and 8GB memory.

3.4.2 Parametric Analysis

We consider the size of seed set $k = 50$ to set the various parameters value of LAPSO-IM under IC diffusion model.

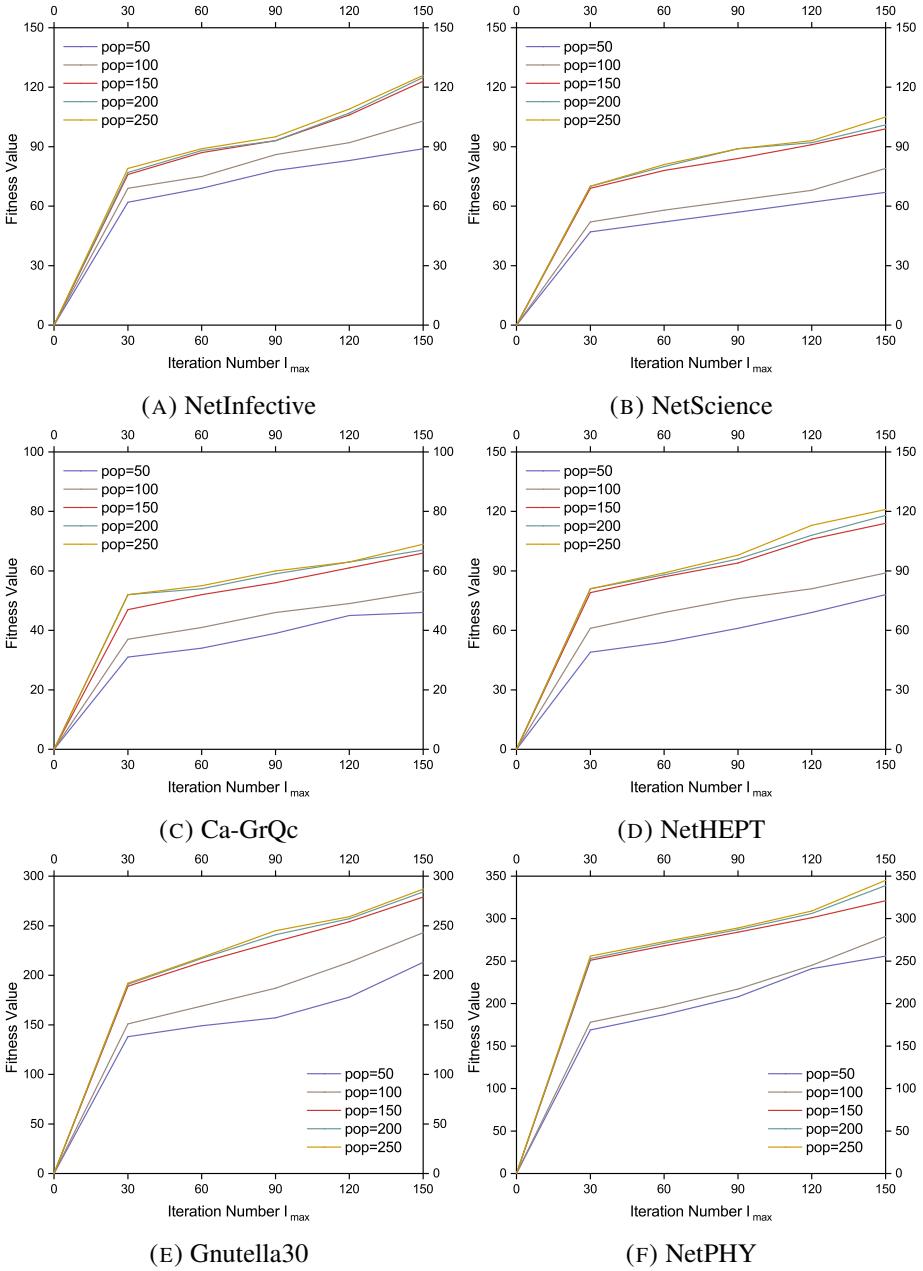
3.4.2.1 The size of the population and the number of iterations

In order to analyze the performance of LAPSO-IM with distinct population size pop and the number of iteration I_{max}, we set learning factors a and b to 1.2, and inertia weight w to 0.8. FIGURE 3.5 and 3.6 show the fitness value of LAPSO-IM with distinct values of I_{max} and pop respectively.

FIGURE 3.5 shows that the fitness values are increased with swarm population pop within 150 iterations. But LAPSO-IM generates very approximate fitness value with $pop > 150$ for six real-world social networks. To achieve a trade-off between running time and performance, we set swarm size $pop = 150$.

FIGURE 3.6 shows the fitness value curves in six real-world networks within 600 iterations. Fitness curves of FIGURE 3.6 illustrate that LADPSO-IM has converged within 150 iterations for all social networks.

In order to achieve a trade-off between running time and performance, $I_{max} = 150$ is a reasonable choice.



(A) NetInfective (B) NetScience

(C) Ca-GrQc (D) NetHEPT

(E) Gnutella30 (F) NetPHY

FIGURE 3.5: The Variation of the Fitness Value on Distinct Swarm Population

3.4.2.2 The inertia weight and learn factors

In order to verify the values of the inertia weight and learn factors, pop and I_{max} are both set to 150. FIGURE 3.7 shows that the fitness values are raised with inertia weight δ for six real-world social networks. But the fitness values with $\delta \geq 0.8$ are very approximate. Thus, we set $\delta = 0.8$ for performance analysis on social networks.

As for the learn factors, we vary the values of learn factors and fix the value of $\delta = 0.8$. As shown in FIGURE 3.8, the fitness values are raised with the increase in learn factors a and b in all six social networks. Therefore, both a and b are set to 1.2.

3.4.2.3 The reward and penalty parameter

In order to set the values of the reward a_r and penalty b_p parameters, we keep the swarm size $pop = 150$, number of iterations $I_{max} = 150$, inertia weight $\delta = 0.8$, learn factors $a = b = 1.2$, and vary the values of reward and penalty parameter between [0.1,0.9]. As shown in FIGURE 3.9, the fitness values are increased with the increase of a_r and b_p. But the fitness values with $a_r \geq 0.6$ and $b_p \geq 0.6$ are very approximate. Thus, we set both a_r and b_p to 0.6.

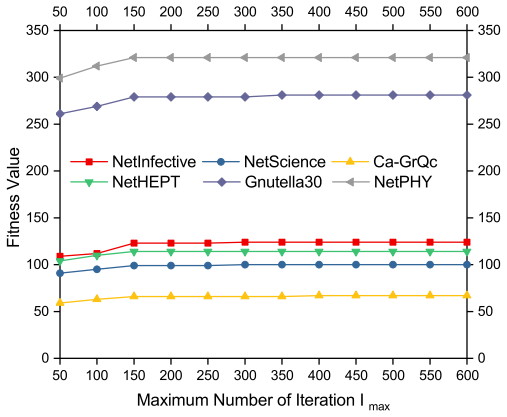


FIGURE 3.6: Iteration Number I_{max}

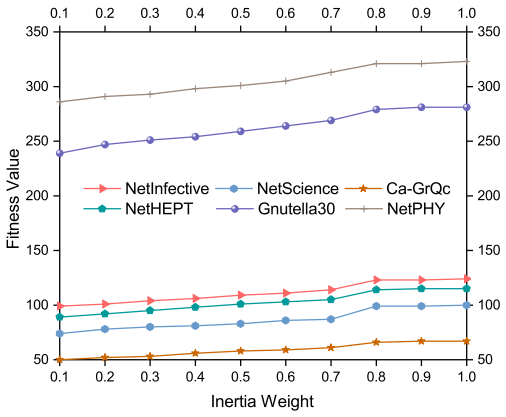


FIGURE 3.7: Inertia Weight δ

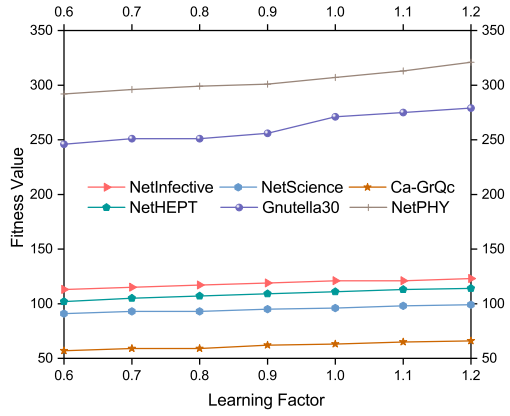


FIGURE 3.8: Learning Factors a and b

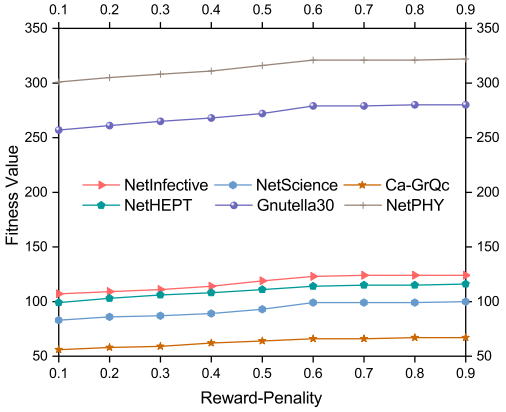


FIGURE 3.9: The Variation of the Fitness Value with Distinct Linear reward-penalty ($a_r = b_p$) Values

3.4.3 Analyzing Quality

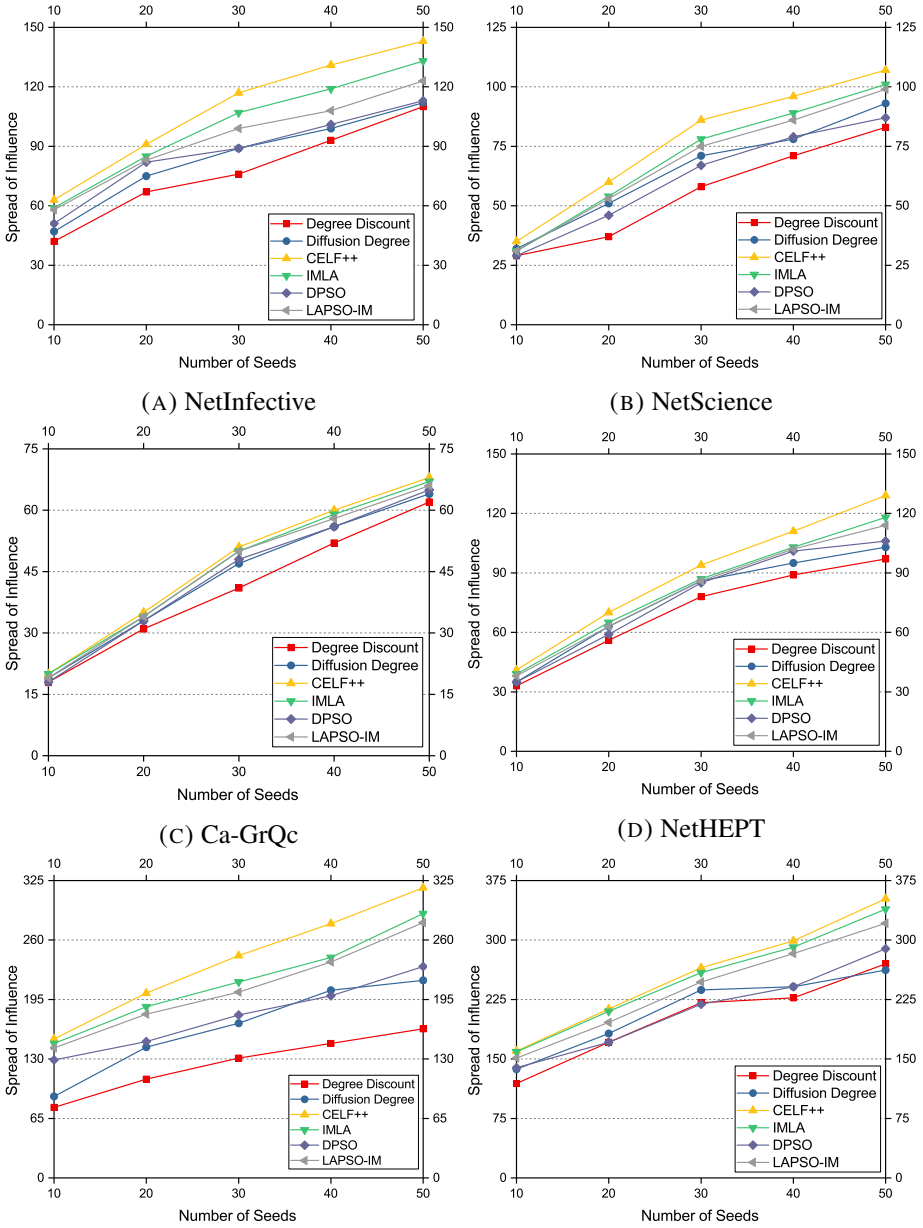
Quality in IM problem equates to the influence spread of seed nodes S in the social network. An algorithm with higher influence spread is known as higher quality algorithm. We select 5 distinct set of seed users of size 10, 20, 30, 40, and 50 for six real-world social networks datasets for quality analysis of LAPSO-IM.

FIGURE 4.6, 3.11, and 4.5 show the influence spread results of LAPSO-IM with the best suited state-of-the-art algorithms under IC, WC,

and LT diffusion models respectively. As shown in FIGURE 4.6, 3.11, and 4.5, LAPSO-IM outperforms both heuristic approaches Degree Discount and Diffusion Degree in terms of influence spread under each diffusion models for all networks. This is because both of heuristic approaches do not consider any feature, context or information except their node degree. LAPSO-IM also performs better than base method DPSO under each diffusion models for all networks. The reason behind this is that DPSO uses only local search strategy, while LAPSO-IM performs local search, global search as well as the combination of both.

LAPSO-IM influence spread is slightly lower than IMLA and CELF++ under each diffusion models for all six real-world social networks. This is because of IMLA and CELF++ both have more stable search strategy with time-consuming MC simulations. Therefore, to acquire a trade-off between influence spread and time, we compromise with the quality of LAPSO-IM in very less to gain high efficiency.

We summarize the influence spread of LAPSO-IM with the state-of-the-art algorithms on each dataset under traditional diffusion models at $S = 50$ as depicted in TABLE 3.2, 3.3, and 3.4. The influence spread is denoted as percentage of nodes in the original network which are influenced by seed set S. As shown in TABLE 3.2, 3.3, and 3.4, the influence spread of LAPSO-IM is better than heuristic techniques and DPSO. The influence spread of LAPSO-IM is quite close to IMLA and CELF++.



(A) NetInfective (B) NetScience

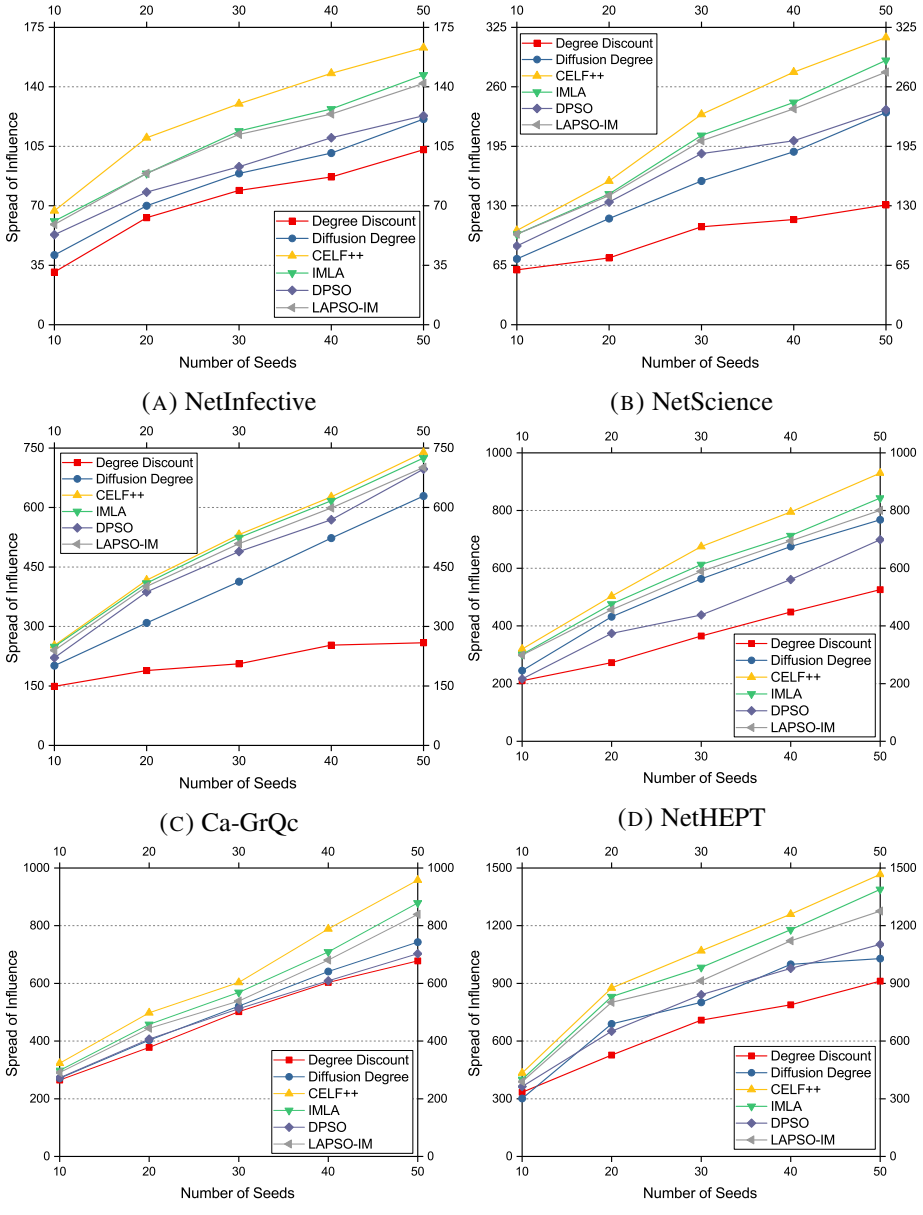
(C) Ca-GrQc (D) NetHEPT

(E) Gnutella30 (F) NetPHY

FIGURE 3.10: The Influence Spread Comparison in Various Datasets under IC Model

3.4.4 Analyzing Efficiency

In this section, we compare the running time of the proposed algorithm with the state-of-the-art algorithms to study how efficiently proposed algorithm finds the distinct size seed set on different datasets. For



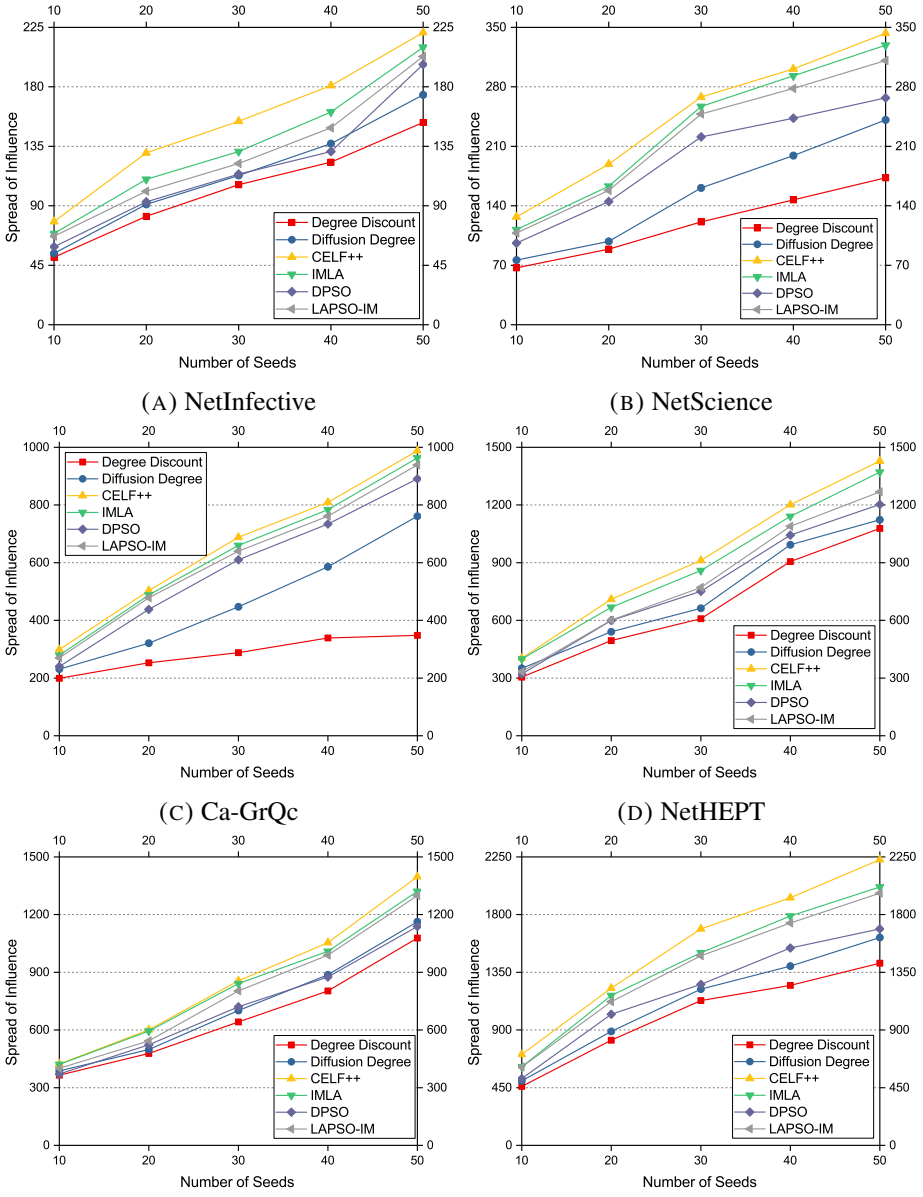
(A) NetInfective (B) NetScience

(C) Ca-GrQc (D) NetHEPT

(E) Gnutella30 (F) NetPHY

FIGURE 3.11: The Influence Spread Comparison in Various Datasets under WC Model

efficiency analysis, we perform the experiment on same size seed set as used in quality analysis of the same six real-world networks. FIGURE 4.8, 3.14, and 4.7 show the comparison of LAPSO-IM with the best suited state-of-the-art algorithms under IC, WC, and LT diffusion models respectively.



(A) NetInfective (B) NetScience

(C) Ca-GrQc (D) NetHEPT

(E) Gnutella30 (F) NetPHY

FIGURE 3.12: The Influence Spread Comparison in Various Datasets under LT Model

As shown in FIGURE 4.8, 3.14, and 4.7, both heuristic approaches Degree Discount and Diffusion Degree are much faster than LAPSO-IM under each diffusion models for all networks. This is because both heuristics consider only node degree to select seed nodes. Heuristic approaches take almost same running time. LAPSO-IM takes almost

TABLE 3.2: The Comparison of Influence Spread (represented in terms of percentage of nodes in original social networks) and Running Time (in terms of seconds) under IC Diffusion Model in Various Datasets at $|S| = 50$

Dataset	Influence Spread						Running Time					
	Degree Discount	Diffusion Discount	CELF++	IMLA	DPSO	LAPSO-IM	Degree Discount	Diffusion Discount	CELF++	IMLA	DPSO	LAPSO-IM
NetInfective	26.83	27.32	34.88	32.44	27.56	30.00	0.02	0.04	2201.09	1598.35	24.19	24.28
NetScience	5.22	5.85	6.73	6.36	5.48	6.23	0.06	0.70	223.43	79.05	26.03	29.02
Ca-GrQc	1.18	1.22	1.30	1.28	1.24	1.25	0.12	0.14	1407.78	817.39	26.01	29.02
NetHEPT	0.64	0.68	0.85	0.77	0.70	0.74	0.35	0.38	3849.59	1835.87	37.94	41.89
Gnutella30	0.44	0.59	0.86	0.79	0.63	0.76	0.35	0.67	5182.30	2103.56	57.89	63.46
NetPHY	0.73	0.71	0.95	0.91	0.78	0.86	1.06	1.29	5487.20	2421.21	64.59	71.32

TABLE 3.3: The Comparison of Influence Spread (represented in terms of percentage of nodes in original social networks) and Running Time (in terms of seconds) under WC Diffusion Model in Various Datasets at $|S| = 50$

Dataset	Influence Spread						Running Time					
	Degree Discount	Diffusion Discount	CELF++	IMLA	DPSO	LAPSO-IM	Degree Discount	Diffusion Discount	CELF++	IMLA	DPSO	LAPSO-IM
NetInfective	29.51	31.71	39.76	35.85	30.00	34.63	0.02	0.04	1901.00	1198.00	18.20	20.28
NetScience	8.24	14.60	19.76	18.19	14.79	17.36	0.06	0.70	171.40	69.10	21.00	24.02
Ca-GrQc	4.94	12.00	14.10	13.83	13.30	13.37	0.12	0.14	1208.00	717.00	24.00	27.02
NetHEPT	3.45	5.04	6.11	5.53	4.59	5.25	0.35	0.38	2850.00	1436.00	27.90	36.89
Gnutella30	1.85	2.03	2.61	2.40	1.92	2.28	0.35	0.67	4182.00	1704.00	47.90	53.46
NetPHY	2.45	2.77	3.95	3.74	2.97	3.43	1.06	1.29	4487.00	1921.00	57.60	64.32

TABLE 3.4: The Comparison of Influence Spread (represented in terms of percentage of nodes in original social networks) and Running Time (in terms of seconds) under LT Diffusion Model in Various Datasets at $|S| = 50$

Dataset	Influence Spread						Running Time					
	Degree Discount	Diffusion Discount	CELF++	IMLA	DPSO	LAPSO-IM	Degree Discount	Diffusion Discount	CELF++	IMLA	DPSO	LAPSO-IM
NetInfective	37.32	42.44	53.90	51.22	48.05	49.58	0.02	0.04	1801.00	1108.00	16.20	20.28
NetScience	10.89	15.17	21.59	20.70	16.80	19.57	0.06	0.70	161.00	61.10	21.00	24.02
Ca-GrQc	6.64	14.52	18.85	18.37	17.00	17.91	0.12	0.14	1108.00	701.00	24.00	27.02
NetHEPT	7.08	7.37	9.38	9.00	7.90	8.32	0.35	0.38	2650.00	1336.00	27.90	36.89
Gnutella30	2.94	3.17	3.81	3.60	3.10	3.54	0.35	0.67	3782.00	1604.00	46.90	53.46
NetPHY	3.82	4.36	6.00	5.43	4.55	5.29	1.06	1.29	4287.00	1821.00	56.60	63.32

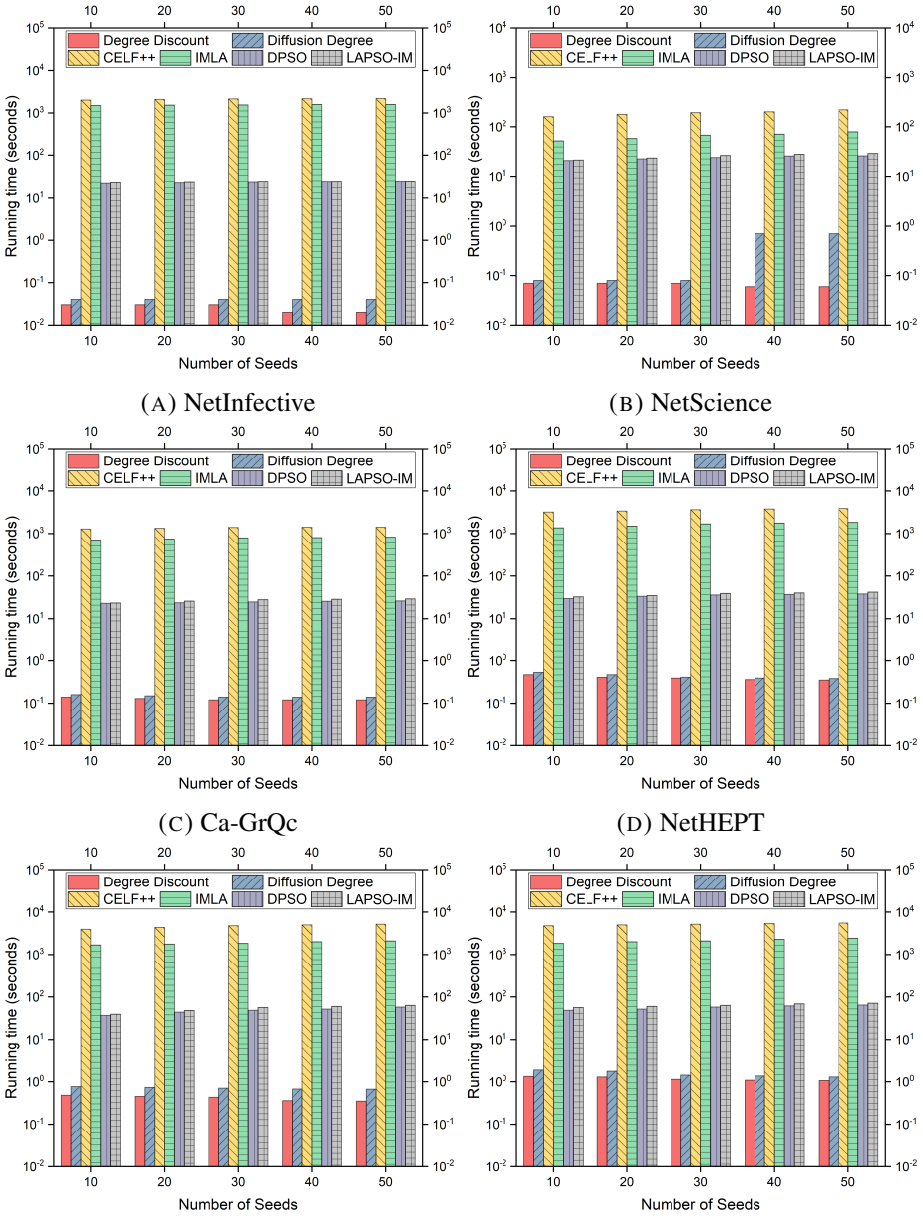
same running time as base method DPSO under each diffusion models for all social networks. LAPSO-IM outperforms both IMLA and CELF++ in terms of efficiency under each diffusion models for all networks. This is because IMLA and CELF++ both have time-consuming MC simulations to simulate influence spread. LAPSO-IM is more efficient than base method IMLA with approximate influence spread and is better than base method DPSO regarding influence spread with comparable running time.

Therefore, LAPSO-IM acquires a trade-off between influence spread and running time.

In TABLE 3.2, 3.3, and 3.4, we summarize the running time (in seconds) of LAPSO-IM with the state-of-the-art algorithms on each dataset under traditional diffusion models at $S = 50$. LAPSO-IM is faster than IMLA and CELF++, and the running time of LAPSO-IM is very similar to base technique DPSO. Therefore, TABLE 3.2, 3.3, and 3.4 show that LAPSO-IM acquires a trade-off between influence spread and time.

TABLE 3.5: The Friedman Test on Average Influence Spread under IC Diffusion Model

Seed	Dataset	IS-value/Rank						Test value	State Result
		Degree Discount	Diffusion Degree	CELF++	IMLA	DPSO	LAPSO-IM	F_f	Is $F_f > \chi^2$?
10	NetInefective	42/1	47/2	63/6	59/5	51/3	58/4	22.3571	Null Hypothesis
	NetScience	29/1.5	32/5	35/6	31/3.5	29/1.5	31/3.5		Rejected
	GrQc	18/1.5	19/3.5	20/5.5	20/5.5	18/1.5	19/3.5		
	NetHEPT	33/1	35/2.5	41/6	39/5	35/2.5	38/4		
	Gnutella30	77/1	89/2	152/6	147/5	129/3	142/4		
	NetPHY	119/1	137/2	160/6	159/5	139/3	151/4		
20	NetInefective	67/1	75/2	91/6	85/5	82/3	83/4	21.5357	Null Hypothesis
	NetScience	37/1	51/2	60/6	54/5	46/3	53/4		Rejected
	GrQc	31/1	33/2	35/6	34/4.5	33/3	34/4.5		
	NetHEPT	56/1	63/3.5	70/6	65/5	59/2	63/3.5		
	Gnutella30	108/1	143/2	202/6	187/5	149/3	179/4		
	NetPHY	171/1	182/3	213/6	210/5	171/2	196/4		
30	NetInefective	76/1	89/2.5	117/6	107/5	89/2.5	99/4	28.9047	Null Hypothesis
	NetScience	58/1	71/3	86/6	78/5	67/2	75/4		Rejected
	GrQc	41/1	47/2	51/6	50/4.5	48/3	50/4.5		
	NetHEPT	78/1	86/3	94/6	87/5	85/2	86/4		
	Gnutella30	131/1	169/2	243/6	214/5	178/3	203/4		
	NetPHY	221/1	237/3	265/6	259/5	229/2	247/4		
40	NetInefective	93/1	99/2	131/6	119/5	101/3	108/4	29.5238	Null Hypothesis
	NetScience	71/1	78/2	96/6	89/5	79/3	86/4		Rejected
	GrQc	52/1	56/2.5	60/6	59/5	56/2.5	58/4		
	NetHEPT	89/1	95/2	111/6	103/5	101/3	102/4		
	Gnutella30	147/1	205/2	278/6	241/5	199/3	236/4		
	NetPHY	227/1	241/2.5	299/6	291/5	241/2.5	283/4		
50	NetInefective	110/1	112/2	143/6	133/5	113/3	123/4	29.5238	Null Hypothesis
	NetScience	83/1	93/3	107/6	101/5	87/2	99/4		Rejected
	GrQc	62/1	64/2	68/6	67/5	65/3	66/4		
	NetHEPT	97/1	103/2	129/6	118/5	106/3	114/4		
	Gnutella30	163/1	216/2	317/6	289/5	231/3	279/4		
	NetPHY	270/1	262/2	352/6	339/5	289/3	321/4		



(A) NetInfective (B) NetScience

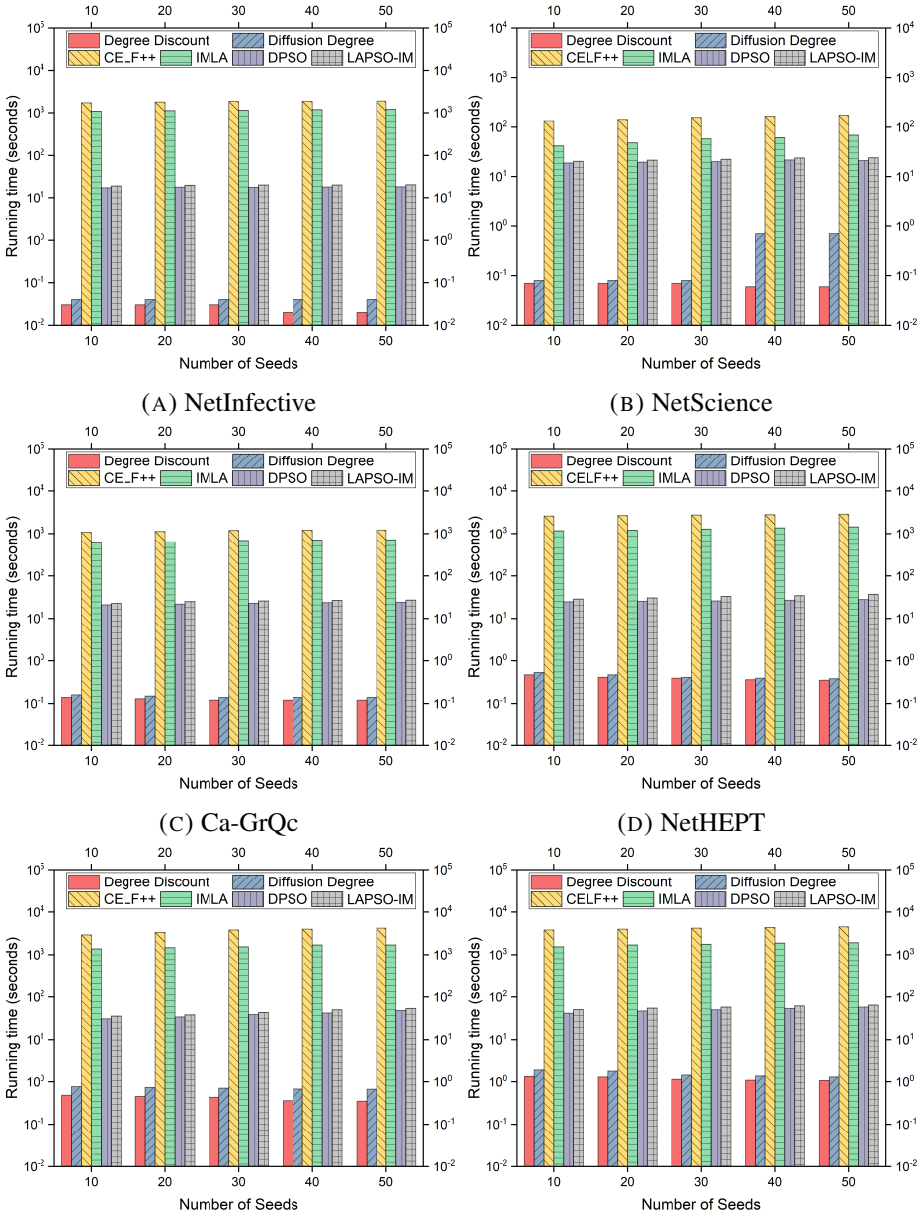
(C) Ca-GrQc (D) NetHEPT

(E) Gnutella30 (F) NetPHY

FIGURE 3.13: The Running Time Comparison in Various Datasets under IC Model

3.4.5 Statistical Test

In this section, we present statistical tests to analyze the significant differences in influence spread between the proposed algorithm and compared algorithms. First, we applied the Friedman test [159, 160] to



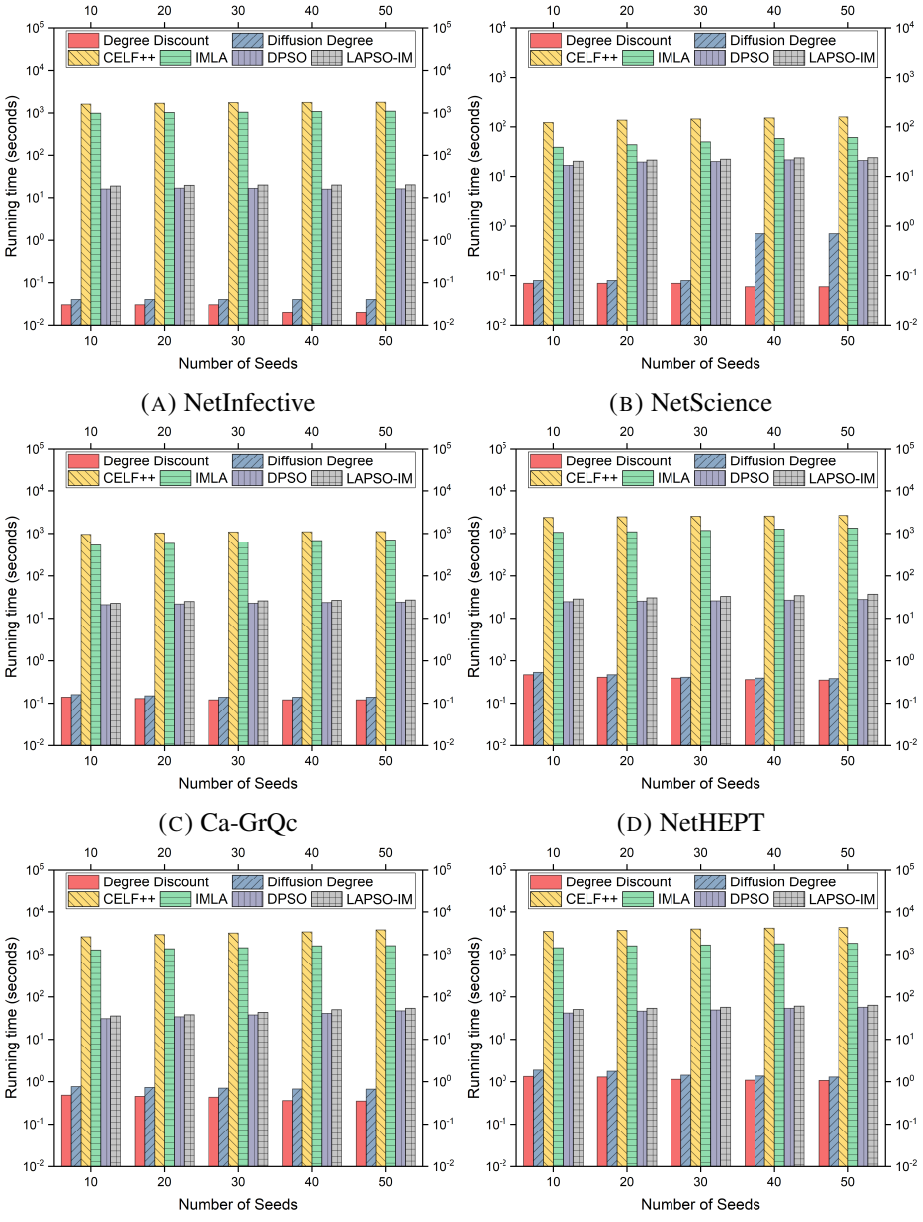
(A) NetInfective (B) NetScience

(C) Ca-GrQc (D) NetHEPT

(E) Gnutella30 (F) NetPHY

FIGURE 3.14: The Running Time Comparison in Various Datasets under WC Model

analyze whether there are significant differences in the average influence spread. If there are significant differences, then we applied the Holland-Bonferroni procedure [160, 161] and the Holm-Bonferroni procedure [160] as post hoc procedure to find the degree of rejection of each hypothesis. Both the post hoc procedures consider LAPSO-IM as



(A) NetInfective (B) NetScience

(C) Ca-GrQc (D) NetHEPT

(E) Gnutella30 (F) NetPHY

FIGURE 3.15: The Running Time Comparison in Various Datasets under LT Model

control algorithm. We consider the level of confidence $\alpha_c = 0.05$ and degree of freedom $D_f = 5$ in all the cases.

The Friedman test indicates that there is a significant difference in average influence spread for each seed set under IC model as shown in TABLE

TABLE 3.6: The Estimation of p-value based on the Holland Procedure for Post-hoc Analysis on Average Influence Spread under IC Diffusion Model

Seed	FRank / z-score / p-value				
	Degree Discount	Diffusion Degree	CELF++	IMLA	DPSO
10	1.16/2.4719/ 0.01344	**2.83/0.9258/ 0.35455**	5.91/1.9257/0.05414	**4.83/0.9258/ 0.35455**	**2.41/1.3146/0.18864**
20	1/2.1571/0.03099	**2.41/0.8517/0.39438**	6/2.4997/0.01243	5/1.5461/0.12208	**2.66/0.6203/0.53506**
30	1/2.8515/0.00435	**2.58/1.3887/0.16492**	6/1.7758/0.07576	**4.91/0.7684/0.44225**	2.41/1.5461/0.12208
40	1/2.7774/0.00548	2.16/1.7033/0.08851	6/1.8516/0.06408	**5/0.9258/0.35455**	**2.83/1.0832/0.27872**
50	1/2.7774/0.00548	2.16/1.7033/0.08851	6/1.8516/0.06408	**5/0.9258/0.35455**	**2.83/1.0832/0.27872**

TABLE 3.7: The Estimation of p-value based on the Holland Procedure for Post-hoc Analysis on Average Influence Spread under WC Diffusion Model

Seed	FRank / z-score / p-value				
	Degree Discount	Diffusion Degree	CELF++	IMLA	DPSO
10	1/2.8515/0.00435	2.25/1.6942/0.09022	6/1.7758/0.07576	**4.91/0.7684/0.44225**	**2.75/1.2313/0.21821**
20	1/2.8515/0.00435	2.25/1.6942/0.09022	6/1.7758/0.07576	**4.91/0.7684/0.44225**	**2.75/1.2313/0.21821/4**
30	1/2.7774/0.00548	2.33/1.5461/ 0.12208	6/1.8516/0.06408	**5/0.9258/ 0.35455**	**2.66/1.2406/0.21475**
40	1/2.7774/0.00548	**2.5/1.3887/0.16492**	6/1.8516/0.06408	**5/0.9258/0.35455**	2.33/1.5461/ 0.12208
50	1/2.7774/0.00548	**2.5/1.3887/0.16492**	6/1.8516/0.06408	**5/0.9258/0.35455**	**2.5/1.3887/0.16492**

TABLE 3.8: The Estimation of p-value based on the Holland Procedure for Post-hoc Analysis on Average Influence Spread under LT Diffusion Model

Seed	FRank / z-score / p-value				
	Degree Discount	Diffusion Degree	CELF++	IMLA	DPSO
10	1/2.7774/0.00548	2.16/1.7033/0.08851	6/1.7758/0.07576	**5/0.9258/0.35455**	**2.83/1.0832/0.27872**
20	1/2.7774/0.00548	2/1.8516/0.06408	6/1.7758/0.07576	5/0.9258/ 0.35455	**3/0.9258/0.35455**
30	1/2.7774/0.00548	2/1.8516/0.06408	6/1.8516/0.06408	**5/0.9258/0.35455**	**3/0.9258/ 0.35455**
40	1/2.7774/0.00548	2.33/1.5461/0.12208	6/1.8516/0.06408	**5/0.9258/0.35455**	**2.66/1.2406/0.21475**
50	1/2.7774/0.00548	2.16/1.7033/0.08851	6/1.8516/0.06408	**5/0.9258/0.35455**	**2.83/1.0832/0.27872**

TABLE 3.9: The Estimation of p-value based on the Holm procedure for Post-hoc Analysis on Average Influence Spread under IC Diffusion Model

Seed	FRank / z-score / p-value / i				
	Degree Discount	Diffusion Degree	CELF++	IMLA	DPSO
10	1.16/2.4719/0.01344	2.83/0.9258/0.35455	5.91/1.9257/0.05414	4.83/0.9258/ 0.35455	2.41/1.3146/0.18864
20	1/2.1571/0.03099	2.41/0.8517/0.39438	6/2.4997/0.01243	5/1.5461/0.12208	2.66/0.6203/0.53506
30	**1/2.8515/0.00435**	2.58/1.3887/0.16492	6/1.7758/0.07576	4.91/0.7684/0.44225	2.41/1.5461/0.12208
40	**1/2.7774/0.00548**	2.16/1.7033/0.08851	6/1.8516/0.06408	5/0.9258/0.35455	2.83/1.0832/0.27872
50	**1/2.7774/0.00548**	2.16/1.7033/0.08851	6/1.8516/0.06408	5/0.9258/0.35455	2.83/1.0832/0.27872

3.5. The null hypothesis (H_0) states that the methods compared are statistically equivalent, with no significant difference. The Friedman test rejects the hypothesis H_0 if the test statistic value F_f is greater than the

TABLE 3.10: The Estimation of p-value based on the Holm Procedure for Post-hoc Analysis on Average Influence Spread under WC Diffusion Model

Seed	FRank / z-score / p-value / i				
	Degree Discount	Diffusion Degree	CELF++	IMLA	DPSO
10	**1/2.8515/0.00435**	2.25/1.6942/0.09022	6/1.7758/0.07576	4.91/0.7684/0.44225	2.75/1.2313/0.21821
20	**1/2.8515/0.00435**	2.25/1.6942/0.09022	6/1.7758/0.07576	4.91/0.7684/0.44225	2.75/1.2313/0.21821
30	**1/2.7774/0.00548**	2.33/1.5461/ 0.12208	6/1.8516/0.06408	5/0.9258/ 0.35455	2.66/1.2406/0.21475
40	**1/2.7774/0.00548**	2.5/1.3887/0.16492	6/1.8516/0.06408	5/0.9258/ 0.35455	2.33/1.5461/ 0.12208
50	**1/2.7774/0.00548**	2.5/1.3887/0.16492	6/1.8516/0.06408	5/0.9258/ 0.35455	2.5/1.3887/0.16492

TABLE 3.11: The Estimation of p-value based on the Holm Procedure for Post-hoc Analysis on Average Influence Spread under LT Diffusion Model

Seed	FRank / z-score / p-value / i				
	Degree Discount	Diffusion Degree	CELF++	IMLA	DPSO
10	**1/2.7774/0.00548**	2.16/1.7033/0.08851	6/1.7758/0.07576	5/0.9258/ 0.35455	2.83/1.0832/0.27872
20	**1/2.7774/0.00548**	2/1.8516/0.06408	6/1.7758/0.07576	5/0.9258/0.35455	3/0.9258/0.35455
30	**1/2.7774/0.00548**	2/1.8516/0.06408	6/1.8516/0.06408	5/0.9258/0.35455	3/0.9258/0.35455
40	**1/2.7774/0.00548**	2.33/1.5461/0.12208	6/1.8516/0.06408	5/0.9258/ 0.35455	2.66/1.2406/0.21475
50	**1/2.7774/0.00548**	2.16/1.7033/0.08851	6/1.8516/0.06408	5/0.9258/0.35455	2.83/1.0832/0.27872

$\chi^2(\alpha_c, D_f)$, i.e. $F_f > 11.0705$. TABLE 3.5 performs the Friedman test for average influence spread and indicates that the null hypothesis is rejected for each k. Similarly, the Friedman test also rejects the null hypothesis for each k under WC and LT models. Therefore, the Holland-Bonferroni procedure (independent test statistics) and the Holm-Bonferroni procedure (dependent test statistics) are applied to measure the concrete differences between the algorithms.

The Holland-Bonferroni procedure rejects hypothesis H_1 to H_{i-1} if p-value is greater than adjusted level of confidence value, i.e., $p_i > 1 - (1 - \alpha_c)^{D_f - i}$, where i is the smallest integer. We consider independent test statistics to perform this post hoc procedure. TABLE 3.6, 3.7, and 3.8 show the p-value/i results for average influence spread under IC, WC, and LT diffusion models respectively. Highlighted values in the

tables indicate rejected hypothesis. LAPSO-IM is used as the control algorithm for post-hoc test.

The Holm-Bonferroni procedure rejects hypothesis H_1 to H_{i-1} if the p-value is greater than the adjusted level of confidence value, i.e., $p_i > \alpha_c/((D_f + 1) - i)$, where i is the smallest integer. We consider dependent test statistics to perform Holm-Bonferroni post hoc procedure. Let $p_1, p_2, \ldots, p_{D_f}$ are the ordered p-values (smallest to largest) and $H_1, H_2, \ldots, H_{D_f}$ are the corresponding hypothesis. The Holm-Bonferroni procedure starts with the most significant p-value. TABLE 3.9, 3.10, and 3.11 show the p-value/i results for average influence spread under IC, WC, and LT diffusion models respectively.

The statistical tests on the IC, WC and LT models demonstrate LAPSO-IM is comparable to both IMLA and CELF++ algorithm, and significantly superior to the remaining three algorithms in terms of influence spread. However, LAPSO-IM outperforms both IMLA and CELF++ in terms of efficiency. Therefore, the proposed algorithm acquires a trade-off between influence spread and efficiency.

3.5 Conclusion

In this chapter, a learning-automata based particle swarm optimization approach is studied and utilized to identify most influential users which maximize influence in social networks. The algorithm maximizes an

objective function that is designed specifically for seed selection in social network using learning-based PSO. The algorithm re-defined the update rule of velocity and position vectors based on each action of learning automata. Proposed algorithm is evaluated based on quality and efficiency against the state-of-the-art algorithms. The analysis revealed following interesting facts.

- The extended local influence evaluation function $EDV^{(2)}(S)$ provides a good trade-off between running time and accuracy to compute the influence spread of a node set.
- The proposed algorithm adopted LA to utilizes the flexible self-adoption and automatic learning capability to learn the budget allocation for each iteration.
- The LA helps avoiding particle's unfruitful moves because it controls the velocity of the particles and regulates particles movement during the search process which enables particles to dynamically search local and global space.
- The empirical results show that the proposed algorithm performs better than DPSO regarding influence spread with almost the same running time. The proposed algorithm outperforms both heuristics Degree Discount and Diffusion Degree in terms of influence spread with low efficiency. LAPSO-IM is more time efficient than both IMLA and CELF++ with approximate influence spread. The

influence spread of LAPSO-IM is better than DPSO with comparable efficiency.

- It is worthwhile to mention that additional LA component has definitely improve convergence of PSO as most of the cases LAPSO-IM is able to reach more nearer to the optimal value.
- The statistical tests are performed to analyze the significant differences in influence spread between the proposed algorithm and compared algorithms. The post hoc analysis states the significant difference between LAPSO-IM and state-of-the-art algorithms.
- In conclusion, LAPSO-IM algorithm obtains a good trade-off between effectiveness and efficiency.