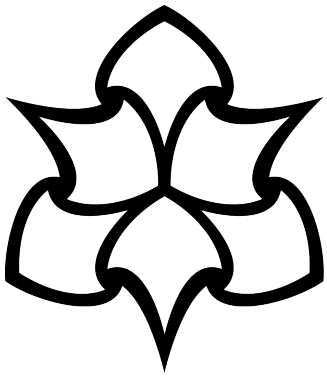


Opinion Leader Detection in Asian Social Networks using Modified Spider Monkey Optimization

SANJAY KUMAR, Department of Computer Science and Engineering, Delhi Technological University, India
AKSHI KUMAR*, Department of Computing and Mathematics, Manchester Metropolitan University, United Kingdom
ABHISHEK MALLIK, Department of Computer Science and Engineering, Delhi Technological University, India
SAKSHI DHALL, Department of Mathematics, Jamia Millia Islamia, India

The Asian social networks are dominated by the society's collectivist culture, and this interestingly introduces a influence mechanism aided by word-of-mouth and opinion leaders. An opinion leader can help to generate and shape other people's opinion and achieve a high information spread on any topic. In this work, a modified spider monkey optimization based opinion leader detection approach is proposed. Firstly, we employ the modified node2vec graph embedding to generate the lower dimensional vectors which act as the initial features for the nodes in a typical Asian social network. Next the entire population is broken down into several groups using the k-means++ algorithm where the number of clusters is equal to the number of opinion leaders to be selected. The local and global leaders are chosen by using the coordinates of the cluster centres of these clusters. The coordinates of the centroids of the clusters are then used to detect the local and global leaders in the network. The local leaders then form the seed set of opinion leaders for the network. The positions of the nodes in the network, including the local and global leaders, are updated over a number of iterations. At the end of these iterations, the seed set generating the maximum influence forms the set of opinion leaders in the network. We test our proposed approach using the popular information diffusion and cognitive opinion dynamics (COD) models. We perform intensive experiments on several real-life social networks based on various performance metrics. The results obtained reveal that the proposed approach outperforms several existing techniques of opinion leader detection.

CCS Concepts: • **Information systems** → **Web applications**; **Social networks**; • **Computing methodologies** → **Machine learning algorithms**.

Additional Key Words and Phrases: Asian Social Networks(ASN), Opinion Leader, Spider Monkey Optimization, k-means clustering

ACM Reference Format:
Sanjay Kumar, Akshi Kumar, Abhishek Mallik, and Sakshi Dhall. 2022. Opinion Leader Detection in Asian Social Networks using Modified Spider Monkey Optimization. In . ACM, New York, NY, USA, 26 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Social networking has a pulsating resplendence in the Asia-Pacific region with the highest social media penetration rates across the globe. In the era of consumerism and with the easy availability of smartphones and connected devices, the social networks in the region have witnessed a high level of engagement. These networks play an important role in generating and mobilizing the public perception regarding any topic. The collectivist behaviours among Asians add to the demographic popularity of various online networks which further stimulates social research with trust and

*Corresponding Author

Manuscript submitted to ACM

influence leadership. There exists numerous research problems in social networks like influence maximization [25], link prediction [3**?**], community detection [23], opinion leader detection. Identifying opinion leaders is a prominent research topic in social networks analysis. Opinion leader is a crucial user with established knowledge proficiency in a particular domain whose opinions are more valued as compared to that of others. Opinion leaders are respected sources of information with sufficient interpersonal communication skills to exert influence on others' decision-making.

Formally, an opinion leader is the user having intensive knowledge in a specific domain such that he can influence the decision, behavior and opinion of other users regarding domain-related topics, ideas or products in the system [1, 5, 10, 35]. Dye [13] defined an opinion leader as a person or a set of persons that have a significant impact on customer's adoption process and decision making as compared to other users. Hence, the opinion leaders play a significant role in generating awareness and shaping public perception about any topic or issues that arise in the society. According to Gold et al. [14] opinion leader detection is a two step process. In the first step, the opinion leader analyzes, examines and understands the end-users requirements and in the second step, opinion leader generates its own opinion based on the knowledge and skills generated from the first step. Due to the large size and evolving nature of social networks, it is difficult to process such large amount of information in an efficient way for problems like finding opinion leaders. Moreover, the general representation of graph using adjacency list or adjacency matrix isn't able to capture the various features of the network in an efficient manner. Network embedding or graph embedding techniques are used to find the lower dimensional vector representations of nodes capturing their topological features in the network. This improves the computational cost of various analysis and mining task performed on these networks.

Over the years, a lot of algorithms have been proposed to optimize the solution set of the opinion leader detection problem. These algorithms aim to arrive at an optimal solution to achieve a higher quality of solution set. In the field of optimization algorithms, one of the most successful class of algorithms are the metaheuristic algorithms. A lot of research has shown that the nature-inspired metaheuristic algorithms have given excellent results for the optimization problems to achieve optimal results [46]. These algorithms develop the solution to an optimization problem by imitating the self-organising or foraging behaviour of animals or colonies appearing in the nature. The primary reason behind the effectiveness of these metaheuristic algorithms is the collaborative behaviour of these algorithms. They use their common knowledge and experience to imporve the solution. The continuous interaction amongst the various agents of the algorithms helps accumulate both the positive and negative feedback in a reward-punishment manner.

In this paper, we tackle the problem of opinion leader detection in social networks. To arrive at a solution we propose a Modified Spider Monkey Algorithm (MSMA) along with graph embedding. We modify the classical spider monkey algorithm [6] to conform with the requirements of social network analysis and suit better to the task of opinion leader detection. This algorithm is based on the foraging behaviour of the spider monkeys. We start off by first generating the modified node2vec [16] graph embedding for every node in the network. These generated embeddings act as the feature vectors for each node which is then used by our Modified Spider Monkey Algorithm. Then we initialize a population of spider monkeys with the nodes in the network being represented as the monkeys and their modified node2vec embeddings acting as the attributes of these monkeys. Then we use k-means++ algorithm to detect a global leader for the entire population. We also use k-means++ to divide the entire population into sub-groups equal to the number of the opinion leaders required. The local leader is assigned to every sub-group by using the cluster centres and these local leaders then go onto become the opinion leaders. The position of each spider monkey is updated based on the collective experience of the global leader, local leader and the monkeys belonging to its local group. The entire process is repeated until the termination condition is satisfied. The set of opinion leaders generated by our algorithm is the set of opinion leaders that generate the optimal influence spread across the various iterations of the algorithm. We investigated the

validity of our proposed algorithm by conducting experiments on eight real-life large scale social networks. We also compared the performance of our algorithm against several classical algorithms and several contemporary metaheuristic based algorithms for opinion leader detection. The experiments were performed under the Independent Cascade (IC), the Susceptible-Infected-Recovered (SIR) information diffusion models. The experimental results reveal the expemplary performance of our proposed algorithm for the task of opinion leader detection by optimally identifying the critical nodes in the network. The major contributions of our work are as follows:

(i) We propose a modification of the classical spider monkey algorithm that can be utilized for the task of social network analysis.
(ii) We propose a nature-inspired metaheuristic based Modified Spider Monkey Algorithm using modified node2vec graph embedding to detect opinion leaders in a social network.
(iii) The Modified Spider Monkey Algorithm uses k-means++ to divide the population into subgroups and obtain local and global leaders for the population.
(iii) We perform exhaustive experimentation on several large scale real-life networks which reveal the commendable performance of our proposed approach for the task of opinion leader detection.

The rest of the paper is organised as follows: Section 2 discusses some of the previously done work in the field of opinion leader detection. Section 3 illustrates some of the preliminary concepts required to gain a better understanding of this work. Section 4 describes our proposed Modified Spider Monkey Algorithm in detail. The description of various real-life datasets and evaluation metrics used by us for performing the experiments is presented in Section 5. The experimental results and evaluation are presented in section 6. Section 7 concludes our work and also mentions the scope of future work.

2 RELATED WORK

The problem of opinion leader detection has been studied in a variety of ways by using differing definitions of the influence of a user in a social network. Various techniques like trust based relationships, friend-foe relationships, game theory, user status, user activities, text-mining, sentiment analysis, node centrality, ontology-based approaches and many more have been employed to detect opinion leader in a social network. Mak et al. [33] proposed a game-theory based approach for influence maximisation. They determined the null or weak associativity results for opinion-leader follower. Goyal et al. [15] presented an approach to detect opinion leaders known as the frequent pattern mining approach. As part of their approach they created a table of user action and analysed their actions. A domain specific opinion leadership approach was hypothesised by Van et al. [42] by linking opinion leaders with social network theory and anticipated that opinion leaders are usually domain specific rather than topic specific. Trusov et al. [41] presented a Bayesian shrinkage in Poisson regression using user's activity records on the premise that a user's activity influences the activities of their friends. A PageRank based approach was proposed by Heidemann et al. [18]. They proposed a quantitative approach by merging together information obtained from user's connectivity and communication activity. They tried their approach on Facebook's New Orleans dataset.

Bodendorf [8] proposed a text-mining based social network specific opinion leader detection approach. They integrated the concepts of social network analysis and text mining to unravel the opinion leaders as well as opinion trends. An ontology-based approach named BARR was propopsed by Li et al [30]. They used information like blog content, authors, readers, and their relationships hence the framework is called BARR for short. Their process is based on first creating ontology around a product and then collecting the relevant data based on BARR. Then they proceed

to identify the disseminators for that data. Cho et al. [12] proposed an opinion leader detection approach based on high sociality using dispersion speed and the supreme cumulative number of adopter. An hybrid framework using a combination of novelty, activity, expertise, and influence evaluated on text contents, time and user behaviour on online learning platform was proposed by Li et al. [31]. Ma et al. [32] proposed a combination of text-mining and network topology based approach to rank super edges in the multi-dimensional model. Aghdam et al. [2] calculated the total trust value for every user in the network. Then they ranked the nodes in the network using their total trust values. Zhu et al. [47] evaluated the emotional preferences of the users using sentiment analysis. Then they created a weighted edge matrix to detect opinion leader. A dynamic social network based approach known as D_OLMiner was presented by Chen et al. [11]. They found the communities and measured the centrality of each user in a dynamic social network.

Most of the above mentioned approaches are purely algorithmic, text-mining, sentiment analysis and ontology-based approaches. Recently some centrality based and metaheuristic based approaches have also been proposed in the recent literature. Yang et al. [45] proposed a novel method based on the closeness amongst the nodes that indicated the relationships amongst various nodes of the network based on different interaction time and delays. The nodes are then ranked based on their closeness values and the nodes with the highest closeness values are considered as the opinion leaders for the networks. Jain et al. [19] proposed a fuzzy logic-based approach in which fuzzy trust rules are derived from fuzzy test systems. They also used the clustering coefficient to ascertain the prominence of the node. This was implemented together to discover the opinion leaders. An innovative social network-based nature-inspired metaheuristic based firefly algorithm was proposed by Jain et al. [20]. They discovered the opinion leaders based on the attractiveness of a node that was computed using the additive centrality of the node in the network. A whale optimization approach was also presented by Jain et al. [21]. They measured the reputation of a user in an online social network based on the various optimization functions. The opinion leader is chosen as the node with the highest reputation.

From a thorough literature survey, it is observed that very little work has been done in the area of opinion leader detection using metaheuristic based approaches. The collaborative nature of the metaheuristic algorithms and their ability to integrate the positive-feedback as well as the negative-feedback of various entities into the collective experience of the entire population makes them a viable option for opinion leader detection. This motivated us towards exploring and proposing a social network based modified spider monkey algorithm for the task of opinion leader detection.

3 PRELIMINARIES

3.1 Graph embedding

Graph embedding is used to condense a graph into a lower dimensional vector space which can be processed easily and efficiently. More formally, it can be stated that these embeddings generate a d dimensional vector for every node in the network. The vectors are oriented in such a way that the nodes which are similar to each other are placed closer together while the nodes which are dissimilar to each other are placed farther away. Some of the recently proposed network embeddings are Laplacian Eigenmaps based embeddings [7] like High-Order Proximity Preserved Embedding (HOPE) [36], Structural Deep Network Embedding (SDNE) [44], etc.

3.1.1 node2vec Graph Embedding. For our study, we have used node2vec graph embedding technique [16]. It is a random walk based embedding technique which uses random walks to better capture the local and global connectivities of the nodes in the network. This gives a clearer picture of how a node is connected in its local and global neighborhood. The node2vec graph embedding works by using a random walk from a vertex (u) and then calculating the probability of

reaching another vertex (v) in that random walk. If the number of times for which the two vertices u and v appear closely in the random walks is high then these vertices are placed closely in the vector space. These generated random walks are then fed into a skip-gram neural network to generate the embeddings for a graph. It has two controlling parameters, p and q. Parameter p defines the probability of discovering the previous node while the parameter q defines the probability of discovering a previously undiscovered node or a previously undiscovered part of the graph. So basically, p controls the local exploration while q controls the global exploration. The basic idea is to make a trade-off between this local and global exploration.

The node2vec embedding method helps in optimally capturing the local and global structure details of the network using the return parameter and the in-out parameter of the random walk. The random walks also help in obtaining a generalised view of the neighborhood of the node under consideration. Moreover, the proposed modification to the classical node2vec algorithm by using the number of neighbors of the node to calculate the unnormalised transition probability gives a better sense of the dominant node in a neighborhood.

3.2 Spider Monkey Algorithm

The Spider Monkey Algorithm [6] is nature-inspired swarm-intelligence algorithm. It relies on the foraging behaviour of the spider monkey in nature. The spider monkeys are a fission–fusion social structure based animals. These animals split themselves into smaller sub-groups and merge themselves into a larger group based on the scarcity or availability of food respectively. Each sub-group has a local leader and the entire population has a global leader. Each spider monkey organises itself according to the position of local leader, global leader and monkeys from their own sub-groups. The positions of the local and global leaders are also changed based on the fitness values of the monkeys in the entire population and in their own groups. Moreover, if a local leader has not changed her position for a certain fixed number of iterations then all the monkeys of that group are redirected in a different direction. While if the global leader has not updated her position for a fixed number of times then she divides the entire group into even smaller sub-groups. If the entire population is already divided into the maximum number of sub-groups and the global leader is still not updating her position, then she combines all the sub-groups into one large group.

3.3 Degree Centrality

Degree centrality [17] assigns an importance score to a node based on the number of edges connected to the node. It works on the simple notion that the more the number of edges associated with a node, the more important it is. The degree of a node u in a network can be expressed as,

$$degree(u) = \mid N(u) \mid \tag{1}$$

here, N(u) represents the set of neighbors of node u.

3.4 Eigenvector Centrality

Eigenvector centrality [17] is used to assign an importance score to the nodes based on the notion that a node connected to high scoring nodes gets assigned a larger score than the nodes that are connected to the same number of less scoring nodes. Mathematically it can be stated as follows:

$$EC(v) = \frac{1}{\lambda} \sum_{u \in N(v), u \neq v} (A_{uv}.EC(u)) \tag{2}$$

here, N(v) represents the set of neighbors of node u. A is the adjacency matrix for the network. λ is a constant defined as follows:

$$A.EC = \lambda.EC \tag{3}$$

3.5 PageRank

PageRank centrality lies at the core of google search engine has become increasing popular since its inception [9]. It builds on the notion of giving importance to nodes receiving more number of links from other nodes. Mathematically it can be expressed as,

$$pagerank(u) = \alpha \sum_{v=1}^{n} \frac{pagerank(v)}{degree_v^{out}} + \frac{1-\alpha}{n} \tag{4}$$

here, α is a damping constant similar to that in katz centrality and $degree_v^{out}$ is the outdegree for the node v.

4 PROPOSED WORK

This section illustrates our proposed work for opinion leader detection in Asian social networks(ASN) in detail. We create context-independent framework to convert the nodes of the network into lower dimensional vector space, and generate the features of the nodes in the network using modified node2vec graph embedding. Thereby, we attain a d dimensional vector representation for every node in the network. This vector represents the topological and structural details of the nodes in the network. These generated vectors then work as the feature vectors for our modified spider monkey algorithm. We modify the traditional spider monkey algorithm to optimize the objective relating to our problem statement. In this modified algorithm, the spider monkeys represent the users in the network while the local leaders form the set of the selected opinion leaders in the network. The various stages of our proposed approach are described as follows.

4.1 Feature Generation using Modified node2vec Embedding

It is often the case that due to limitations in data mining techniques, only several or sometimes none of the nodes have attributes associated with them. Moreover, even when these attributes are present, they are context-specific and thus lead to biased data analysis. Therefore, we aim to present a context-free opinion leader detection technique and for that we don't use the default attributes associated with the nodes of the network. Instead, we generate our own features based on the topological and structural details of the network. Hence, we present a modified node2vec graph embedding to generate d dimensional vector representations for the nodes. This helps to represent the nodes in a lower dimensional vector space capturing the network connectivities while remaining computationally efficient.

The proposed modified node2vec embedding generation has three phases, namely, computing transition probabilities using preprocessing, simulating random walks, and finally optimizing using Stochastic Gradient Descent (SGD). It uses return parameter, p and in-out parameter, q to guide the random walks. The return parameter (p), dictates the probability of revisiting a node immediately in the random walk. The in-out parameter (q), makes the search more localised or globalised. Let a random walk go from node u to node v and now it has to decide where to go from node v. Let the next node that the walk goes to be s. The search bias $\alpha_{pq}(u, s)$ helps to determine the unnormalised transition probability and is represented by Eq. 5.

$$\alpha_{pq}(u,s) = \begin{cases} \frac{1}{p}, & d_{us} = 0 \\ 1, & d_{us} = 1 \\ \frac{1}{q}, & d_{us} = 2 \end{cases} \tag{5}$$

Here, d_{us} represents the shortest path between the nodes u and s. The value of d_{us} can only be either 0, 1, or 2. Using the search bias represented in Eq. 5, the unnormalised transition probability can be calculated using Eq. 6. It represents the probability of choosing a node s as the next step in the random walk. Here, w_{vs} represents the weight of the edge (v,s). For unweighted graphs, the value of w_{vs} is 1.

$$\pi_{vs} = \alpha_{pq}(u,s) \cdot w_{vs} \tag{6}$$

Since, this work focuses on detecting opinion leaders and some of the network datasets available to us are unweighted, so using the weight of the edge w_{vs} isn't very relevant. Hence as part of our proposed modification, we use the total of neighbors of the node s to calculate the unnormalised transition probability. This helps in increasing the transition probability of the node which has more neighbors. It can be represented by Eq. 7. Here, $N(s)$ represents the total number of neighbors of the node s.

$$\pi_{vs} = \alpha_{pq}(u,s) \cdot N(s) \tag{7}$$

The next node in the random walks are selected based on Eq. 8. Here, c_i denotes the i^{th} node in the random walk, π_{vs} is the unnormalised transition probability in going from node v to node s, Z is the normalisation constant and is equals to the total number of transition probabilities and E is the set of all the edges. The generated random walks are then trained in the Skip-gram model in word2vec and the Stochastic Gradient Descent (SGD) algorithm and the embedding for each node is generated. Generated embeddings thereby form the d dimensional feature vectors for the nodes in the network. These generated feature vectors are then utilized by our modified spider monkey algorithm to detect opinion leaders in the network.

$$P(c_i = s \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vs}}{Z}, & (v,s) \in E \\ 0, otherwise \end{cases} \tag{8}$$

4.2 Modified Spider Monkey Algorithm

In this section, we describe our modified spider monkey algorithm. We modify the classical spider monkey algorithm to better suit the problem of opinion leader detection. Moreover, the classical spider monkey algorithm is used for continuous optimization problems whilst we work on selecting several opinion leaders given the initial population. In our proposed modification, we have modified the traditional spider monkey algorithm to better fit the opinion leader detection algorithm in social networks. We have chosen the spider monkey algorithm as it works on the principle of fission-fusion social structure amongst the animals. This divides the entire population into several sub-groups. Each sub-group has its own local leader and the entire population has a global leader as well. All the monkeys align their positions based on their local and global leaders. The concept of local leaders is utilised by us to select the opinion leaders in the sub-groups as they can better disseminate the information in their own group. This is often the case in the nature and amongst the various social networks that not every individual adheres to the opinion of the global leader but the opinion of the local leaders can still be used to motivate the majority of the population in the same direction.

The global and local leaders for our network are selected using the k-means++ algorithm. The various phases of our proposed modified spider monkey algorithm are described as follows:

4.2.1 Initialization. We initialise our network dataset to conform with the concept of monkeys to fit the proposed algorithm. Hence, we treat the nodes of the network as the monkeys. We generate a d dimensional vector for every node in the network. The vectors are generated using modified node2vec graph embedding as described in section 4.1. These vectors then form the feature vectors associated with every node in the network. After the initialization process, we have n ($n =\mid V \mid$ the set of nodes in the network) spider monkeys each attributed by a d dimensional vector space and represented as follows:

$$SM_i = node2vec(i),\ i \in V \& \mid SM_i \mid = d \tag{9}$$

Where, SM_i represents the i^{th} spider monkey in the population.

4.2.2 Global Leader Learning Phase. In this phase, we select a global leader for the entire network by evaluating the suitability of every node based on its feature vector. We use k-means++ clustering algorithm to detect the global leader, which helps in dividing the entire population into several sub-groups or clusters which is the crux of the spider monkey algorithm. By using k-means++ algorithm, we can map the desired number of opinion leaders to the centroids of the clusters. So by varying the value of $'k'$ in the algorithm, we can choose different number of opinion leaders (k) and predict the overall opinion spread in the network to assess the effectiveness of proposed algorithm.

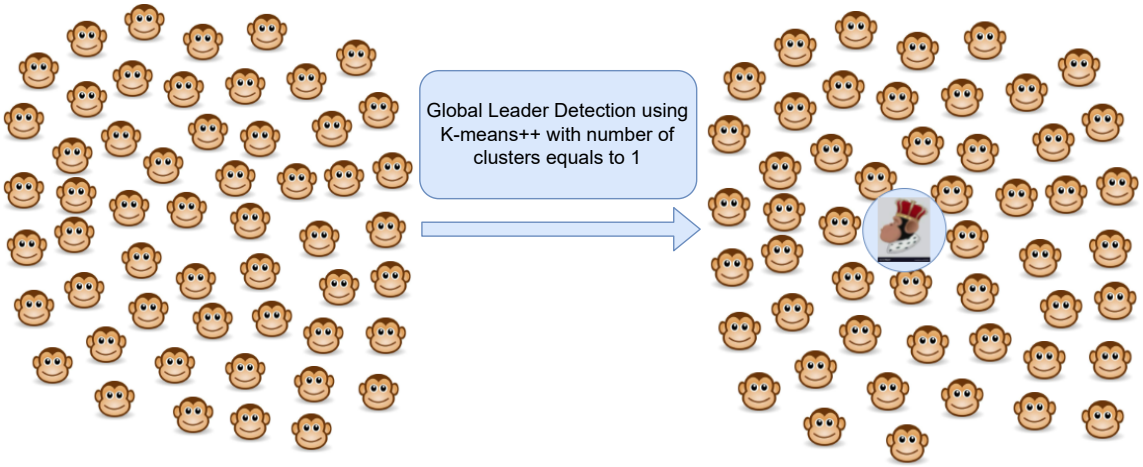


Fig. 1. Global Leader Learning Phase.

Initially, we work by assuming the entire network as a single cluster and obtain the centre of this cluster. The global leader is the spider monkey or the node closest to the centre of the aforementioned cluster. This is described in algorithm 1. Steps 1 and 2 are used to train a k-means++ model and extract the cluster centre coordinates respectively. Steps 3 to 10 generate a list, *Distances* that stores the information about the node and its distance from the cluster centre. V in step 4 represents the set of nodes in the network while d represents the chosen dimension of the graph embedding. Step 11 sorts the *Distances* list according to the distance of every node from the cluster centre in an ascending manner.

Step 12 extracts the global leader finally returns it. Fig. 1 shows the procedure of selecting the global leader from a swarm of monkeys.

Algorithm 1 Algorithm for Global Leader Learning Phase

Input: Nodes of the network represented in d dimensional vector space, G
Output: Global leader
1. Model ← k-means++(G, number_of_clusters = 1)
2. Cluster ← Model.Cluster()
3. Distances := []
4. For i in range(1, V):
5. dist := 0
6. For j in range (1, d):
7. dist ← dist + sqrt(SM_{ij}^2-$Cluster_{ij}^2$)
8. end For
9. Distances.insert({i, dist})
10. end For
11. Distances ← sort_by_second(Distances)
12. Global_Leader ← Distances[0][0]
return Global_Leader

4.2.3 Local Leader Learning Phase. This section describes the modifications done in the local leader learning phase of the traditional spider monkey algorithm. Contrary to the classical heuristic, we don't select the local leader using a greedy approach. But instead, we use the k-means++ algorithm for selecting the local leader as we did in the global leader learning phase 4.2.2. Since the concept of local leader refers to splitting the entire population into smaller sub-groups on the basis of the similarities amongst the monkeys or nodes in the network. Therefore, we utilize the k-means++ algorithm as it splits the entire population into the desired number of clusters and also generates the cluster centres which we utilize to decide the local leaders. As the opinion leader detection problem refers to selecting k leader nodes from the entire population, so we split the entire population into k clusters and obtain the cluster centres for every cluster. The local leader of a particular cluster is the node closest to the cluster centre of that cluster. This is described in algorithm 2. Similar to algorithm 1, we first train the k-means model and extract the cluster centres in step 1 and 2. Then in step 3, we initialize a *Local_Leaders* list to store the local leaders for every sub-group in the network. Then in steps 4 to 15, we generate the local leaders for every sub-group similar to the way we generated the global leaders in algorithm 1. Finally we return a list of local leaders. Fig. 2 represents the process of selecting the local leaders by dividing the entire swarm into small clusters or subgroups.

4.2.4 Storing the Opinion Leaders. Since the major task of the opinion leaders is to simulate the maximum information diffusion across the network. So an optimal set of opinion leaders is the one which disseminates a specific opinion in an efficient and effective manner under a chosen information diffusion model (*IDM*). In our modified spider monkey algorithm, we consider the selected local leaders to be the opinion leaders in the network. So, we run our algorithm for several iterations and for every iteration, we store the selected opinion leaders and the influence spread achieved by them. After all the iterations, we select the set of opinion leaders which performed best in terms of influence spread.

4.2.5 Local Leader Phase. This phase updates the position of every spider monkey in the d dimensional vector space using the position of the local leader and a random member of the group to which the monkey belongs to. This helps

Algorithm 2 Algorithm for Local Leader Learning Phase

Input: Nodes of the network represented in d dimensional vector space, G and number of desired opinion leaders, k
Output: Local leaders

```
1. Model ← k-means++(G, number_of_clusters = k)
2. Clusters ← Model.Clusters()
3. Local_Leaders := []
4. For g range(1, k):
5.    Distances := []
6.    For each member SM_i ∈ g^th group:
7.       dist := 0
8.       For j in range (1, d):
9.          dist ← dist + sqrt(SM_ij^2-Cluster_gj^2)
10.       end For
11.       Distances.insert({i, dist})
12.   end For
13.   Distances ← sort_by_second(Distances)
14.   Local_Leaders[g] ← Distances[0][0]
15. end For
   return Local_Leaders
```

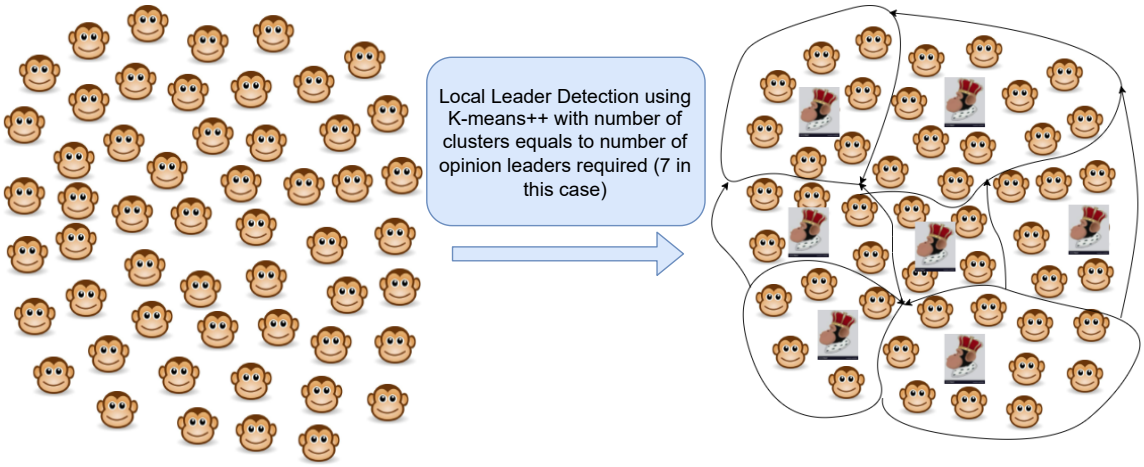


Fig. 2. Local Leader Learning Phase.

the monkeys or the users to update their positions based on the collective experience of the group. The position of the ith spider monkey belonging to the k^{th} group is updated as follows:

$$SM_{newij} = SM_{ij} + U(0,1) \times (LL_j - SM_{ij}) + U(-1,1) \times (SM_{rj} - SM_{ij}) \tag{10}$$

where SM_{ij} is the j^{th} dimension of the i^{th} spider monkey. LL_j represents the j^{th} dimension of k^{th} local group's leader. SM_rj represents the j^th dimension of a random monkey chosen from the k^{th} group such that r≠i. U(l,e) is a uniformily distributed random number between l and e. Algorithm 3 illustrates the local leader phase in detail.

Algorithm 3 Algorithm for Local Leader Phase

Input: Nodes of the network represented in d dimensional vector space, G and number of desired opinion leaders, k
Output: Position updated spider monkeys, SM_{newij}

1. For g in range(1, k):
2. For each member $SM_i \in g^{th}$ group:
3. r ← random monkey in g^{th} group with r≠i
4. LL ← Local_Leaders[g]
5. For j in range(1,d):
6. SM_{newij} = SM_{ij}+U(0,1)×(LL_j-SM_{ij})+U(-1,1)×(SM_{rj}-SM_{ij})
7. end For
8. end For
9. end For

return SM_{newij}

4.2.6 Global Leader Phase. In the global leader phase also, we update the position of the spider monkeys. In this phase every spider monkey updates its position based on the position of the global leader and the local members of the group, the monkey is a member of. The positions of the monkeys or users in the entire network are updated as follows.

$$SM_{newij} = SM_{ij} + U(0,1) \times (GL_j - SM_{ij}) + U(-1,1) \times (SM_{rj} - SM_{ij}) \tag{11}$$

where SM_{ij}, SM_{rj} and U(l,e) are same as described in section 4.2.5. While G_j represents the j^{th} of the global leader. This phase is further described in detail in Algorithm 4.

Algorithm 4 Algorithm for Global Leader Phase

Input: Nodes of the network represented in d dimensional vector space, G and number of desired opinion leaders, k
Output: Position updated spider monkeys, SM_{newij}

1. For g in range(1, k):
2. For each member $SM_i \in g^{th}$ group:
3. r ← random monkey in g^{th} group with r≠i
4. For j in range(1,d):
5. SM_{newij} = SM_{ij}+U(0,1)×(GL_j-SM_{ij})+U(-1,1)×(SM_{rj}-SM_{ij})
6. end For
7. end For
8. end For

return SM_{newij}

4.2.7 Termination. Any metaheuristic algorithm is associated with a termination condition. It refers to the condition which when satisfied terminates the algorithm. The termination condition is algorithm and problem specific. We also define the termination condition according to the opinion leader detection problem. Due to the lack of ground truth regarding the opinion leaders, there is no way to find out whether we obtained the best set of opinion leaders. Hence, we run our algorithm for a certain number of fixed iterations. The set of opinion leaders and their achieved performance in terms of influence spread is stored for every iteration as described in section 4.2.4. Once we have run our algorithm

for all the iterations, we select the set of opinion leaders with the best performance in terms of influence spread as our set of opinion leaders.

The flow diagram and various steps of our proposed modified spider monkey algorithm is illustrated in Fig. 3. This flowchart captures the various steps described in the previous sections. It can be seen from Fig. 3 that our algorithm first generates the embedding using the modified node2vec for every node in the network. Then it goes on to initialize a population comprising of the node or users in the networks as spider monkeys with their modified node2vec embeddings as the features of the spider monkeys. Then we use k-means++ algorithms to select global leader (using 1 cluster) and local leaders (using 'K' clusters). The local leaders are then chosen as the opinion leaders and the performance of these opinion leaders in terms of influence spread is calculated and stored. Then if the termination condition is not satisfied then our algorithm updates the position of every spider monkey based on the positions of global and local leaders. It can also be seen from Fig. 3 that how our proposed approach develops on an efficient solution iteratively. It also shows that as soon as the termination condition is met, it evaluates the best solution of opinion leaders and terminates.

4.3 Computational Time analysis

In this section, we estimate the computational time of our proposed Modified Spider Monkey Algorithm (MSMA). The time complexity analysis is divided into four phases: The feature generation phase, the local and global leader detection phase, the position updation phase and finally the opinion spread calculation phase. Let us consider a network $G(V, E)$ where V is the set of nodes and E is the set of edges where $\mid V \mid$ = n, and $\mid E \mid$ = m. For the first phase, we generate features using the modified node2vec embedding as shown in section 4.1. The time complexity for this phase is $O(mlogm)$ [37]. Then we go on to selecting the global and local leaders using the K-means++ algorithm as shown in section 4.2.2 and 4.2.3 respectively. The time complexity for the K-means++ algorithm turns out to be $O(nkd)$ [4], where n is the number of data points, k is the number of clusters and d is the dimensionality of the data. In our case the number of data points is the number of nodes in the network. The number of clusters, k which is 1 while detecting the global leaders and number of opinion leaders required while detecting the local leaders. The dimensionality of the data is the length of the feature vector generated by us in the first phase. The dimensionality, d and the number of clusters, k is usually a very small constant value as compared to the number of nodes in the network. Hence the time complexity for the second phase is $O(nkd) \simeq O(n)$.

Then in the third phase, we go onto updating the position of every monkey using the Local leader phase and global leader phase as shown in Section 4.2.5 and 4.2.6 respectively. In the local leader phase, we update the position of every spider monkey based on the local leader and a random member of the group to which the monkey belongs. Since we update every dimension of the position of the monkey and we do this for every monkey, so the time complexity for local leader phase is $O(nd)$. For the global leader phase, we update the position of every monkey based on the position of the global leader and a random member of the group to which the monkey belongs. This can be done with a time complexity of $O(nd)$, since here also we update every dimension of the position of every monkey. Hence the combined time complexity of the third phase is $O(nd + nd) = O(nd)$. Since d is a very small constant compared to n, so the time complexity of third phase is $O(n)$.

In the fourth phase, we calculate the influence spread of the chosen opinion leaders. Considering the case of SIR model, we find the total number of infected nodes at the end of the simulations instigated by the set of seed nodes. For the SIR model, once a node gets infected then it can infect its neighboring node with a probability of β, known as infection probability. In this study, we take β to be 0.1. To compute the spreading influence of all nodes using SIR model requires $\beta \times \sum_{v_i \in V} d_i \simeq \beta \times O(m + n) \simeq O(m + n)$ time, i.e., sum of the degree of all the nodes in a network is equal to

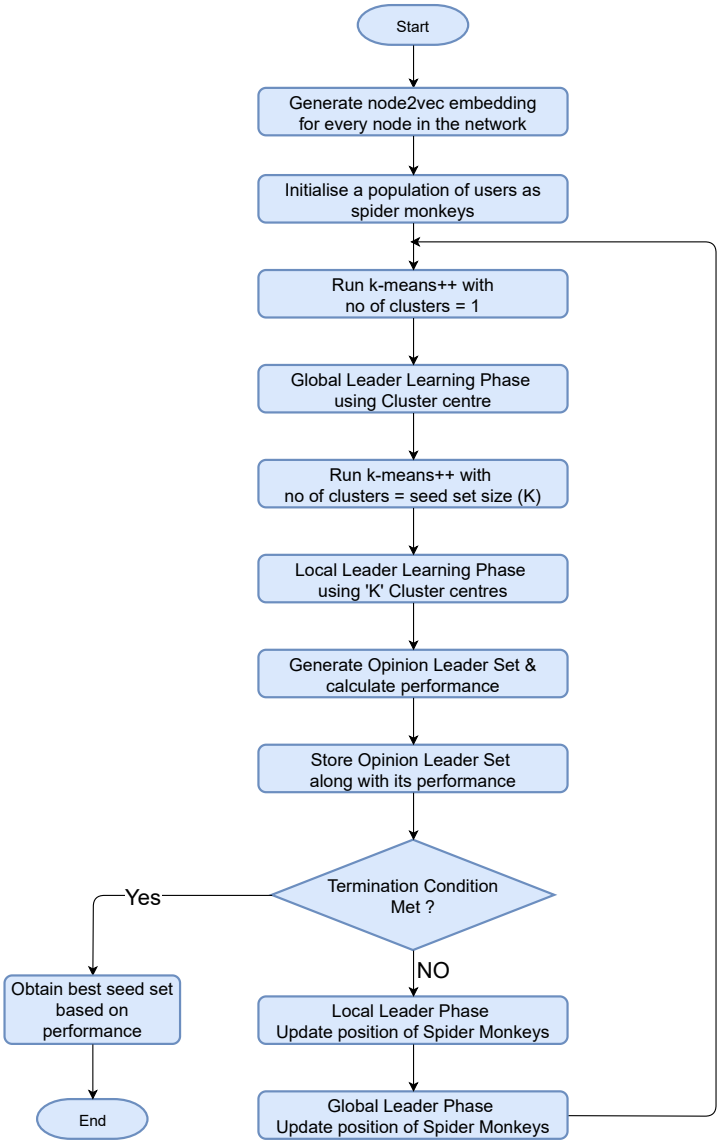


Fig. 3. Algorithmic flow chart of our proposed modified Spider Monkey Algorithm for Opinion Leader Detection.

the addition of total number of nodes and edges present in the system where d_i denotes the degree of node v_i and β is a constant. Therefore, the time complexity for the third phase is $(n + m)$. We perform the second phase and the third phase a total of $I(maximumnumberofiterations)$ times. Hence the time complexity for the combined run of second, third and fourth phase becomes $O(I(n + n + n + m))$. Since I is a small fixed constant as compare to the large number of nodes and edges, this time complexity becomes $O(I(n + n + n + m)) \simeq O(n + m)$. Hence, the total time complexity for our proposed approach is $O(mlog(m) + n + m)$. In general, the social networks are sparse networks, we can assume that $O(m) = O(n)$. Finally, the time complexity of our proposed algorithm is $O(nlogn)$.

5 DATASETS AND EVALUATION METRICS

This section presents the various datasets and evaluation metrics chosen by us for our study. The datasets chosen belong to varying domains and are sampled from several online social networks. The evaluation metrics on the other hand help us to better evaluate the quality of the opinion leader set generated by our algorithm.

5.1 Dataset

This section describes the datasets chosen by us. We choose eight online social networks of varying context, size and complexities. Tab. 1 tabulates the statistical details of the networks used in our work. The description of the datasets used by us is as follows:

(i) LastFM [40]:
It represents the LastFM network of Asian users. Nodes represent the LastFM users while the edges represent the relationships between them.

(ii) Epinions [38]:
This network represents the data collected from the consumer review site Epinions.com. It is a who-trust-whom online social network.

(iii) Wiki [27, 28]:
It is the voting network representing the votes casted by users to promote an existing user to the position of administrator.

(iv) Bitcoin [24, 26]:
This is a who-trust-whom network of the users trading bitcoin on the bitcoin trading platform Bitcoin OTC.

(v) Github [39]:
It is a network of developers on an online social network for developers, Github. Nodes are developers who have starred at least ten repositories on Github and edges represent the existence of a follower-followee relationship amongst the nodes.

(vi) Deezer [40]:
This network represents the Deezer users across Europe. The nodes are the users in the network while the edges represent the follower relationships amongst the nodes.

(vii) Facebook [34]:
This network comprises of the circles or the list of friends from Facebook. The data was collected from survey participants using the Facebook app.

(viii) Email [22, 29]:
It is an email communication network from Enron. The nodes represents the users while the edges represent the existence of email communication amongst the nodes.

Table 1. The basic topological features of the networks used. Here, $|V|$, $|E|$, d_{max}, d_{min}, and d_{avg} reprsents number of vertices, number of edges, maximum degrre, minimum degree, and average degree in the network, respectively.

Datasets	| **V** |	| **E** |	Type	d_{max}	d_{min}	d_{avg}
LastFM	7624	27806	Undirected	98	1	6
Epinions	75879	508837	Directed	443	1	7
Wiki	7115	103689	Directed	102	1	6
Bitcoin	5881	35592	Directed	496	1	8
Github	37700	289003	Undirected	3700	1	7
Deezer	28281	92752	Undirected	112	1	6
Facebook	4039	88234	Undirected	419	1	43
Email	36692	183831	Undirected	1400	1	10

5.2 Evaluation Metrics

This section describes the performance standards evaluated by us for ascertaining the performance prowess of our algorithm in detecting the opinion leaders. The details of these performance metrics are given below:

5.2.1 Total Opinion Spread. We evaluate the performance of our proposed model, we compare the total opinion spread of the selected opinion leaders for various algorithms. Total opinion spread can be defined as the cumulative sum of the value of opinions possessed by every node in the graph. Initially a few opinion leaders try to disseminate a particular opinion across the network. Every node in the network thereby absorbs only a fraction of that opinion depending upon the network topology. This metric is modelled for the Cognitive Opinion Dynamics model introduced by Vilone et al. [43]. The opinion of an individual can take values from 0 to 1. 1 being an alarmist opinion while 0 being a non-alarmist opinion. Initially, the opinion of the opinion leaders is modelled as 1.

5.2.2 Influence Spread. To evaluate the performance of our proposed model, we also compare the achieved influence spread of the selected opinion leaders. The selected opinion leaders form the seed set of size k which is initially activated under a desired information propagation model. Given a social network G, a seed set of opinion leaders S and an information diffusion model, then the influence spread σ(S) of the set S is the total number of seed nodes and the nodes that are activated during the spreading process under the given information diffusion model. We evaluate the influence spread achieved by our algorithm by varying several parameters as follows:

(i) Influence Spread vs. Number of Opinion Leaders, σ(S) vs. k:
It is evident that the influence spread tends to increase as the number of opinion leaders chosen increases. So we vary the size of the seed set (number of opinion leaders) to ascertain its impact on the influence spread achieved by the algorithm under consideration under different information diffusion models.

(ii) Influence Spread vs. Infection Probability:
For the IC and the SIR information diffusion models, one of the major influence control parameters is the infection probability, p and β, respectively. As the infection probability increases the influence spread also increases. Hence,

we evaluate the effect of varying the infection probability, p and β for IC and SIR model on the influence spread achieved by the algorithm under consideration.

6 EXPERIMENTAL RESULTS AND ANALYSIS

This section presents and discusses the experimental setup and the experimental results obtained by us. We tested our proposed opinion leader detection method on an Asian social network, namely, LastFM and then we further generalised our approach to other social networks. In total, we have performed the experimentation on eight real-life datasets mentioned in section 5.1 and evaluated all the performance metrics mentioned in section 5.2. Intensive experiments were carried to understand the performance capabilities of our proposed approach, Modified Spider Monkey Algorithm (MSMA). The performance of our algorithm was also compared with several classical algorithms like Degree Centrality (DC) [17], Eigenvector Centrality (EC) [17], PageRank Centrality (PR) [9]. Since our proposed work is a meta heuristic based algorithm so we compare its performance with some metaheuristic based opinion leader detection techniques as well like Firefly [20] and Whale optimization [21]. All the results were obtained for the Independent Cascade (IC), the Susceptible-Infected-Recovered (SIR) As all these models are stochastic in nature, so we perform all the experiments 100 times and the obtained results are averaged out for every model. This helps to achieve more reliable results.

6.1 Total Opinion Spread vs. Number of Opinion Leaders

In this section, we compare the total opinion spread achieved by various algorithms as the number of opinion leaders are varied. We evaluate this metric for the Cognitive Opinion Dynamics (COD) based information diffusion model. We ran this model for a total of 100 iterations for every algorithm and the results of every iteration were then averaged out. Fig. 4 presents the results obtained for this metric for all the datasets and for all the chosen algorithms. As can be seen from fig. 4 that with an increase in the number of opinion leaders, the total opinion spread also increases for every network. For networks like, Epinions (Fig. 4(b)), Wiki (Fig. 4(c)), Bitcoin (Fig. 4(d)) and Github (Fig. 4(e)) the difference between the total opinion spread achieved by our algorithm and other algorithms is immense. This shows that our proposed work is a clear winner in these networks. While for the networks like Facebook (Fig. 4(a)), Deezer (Fig. 4(f)), LastFM (Fig. 4(g)) and Email (Fig. 4(h)), there tends to be a stiff competition amongst the various algorithms. But even for such networks, our algorithm performs better than other. This validates the utility of our proposed Modified Spider Monkey Algorithm (MSMA) for the task of spreading a particular opinion across the networks.

6.2 Influence Spread vs. Number of Opinion Leaders, σ(S) vs. k

In this section, we present the influence spread attained by our proposed Modified Spider Monkey Algorithm (MSMA) and various classical and contemporary algorithms for opinion leader detection as the number of opinion leaders is varied. The results are obtained on all the real-life datasets mentioned in section 5.1 and for IC, and the SIR information diffusion model. The particular results achieved for each information diffusion model is described below:

(i) Independent Cascade (IC) Model:

Fig. 5 shows the influence spread achieved by various algorithms as the number of opinion leaders is varied. The infection probability, p for the IC model is set at 0.25 for all the datasets. As can be seen from fig. 5 that the influence spread increases as the number of opinion leaders increases. This can be attributed to the fact that the as the number of initially infected population increases, the number of activated users would also during the

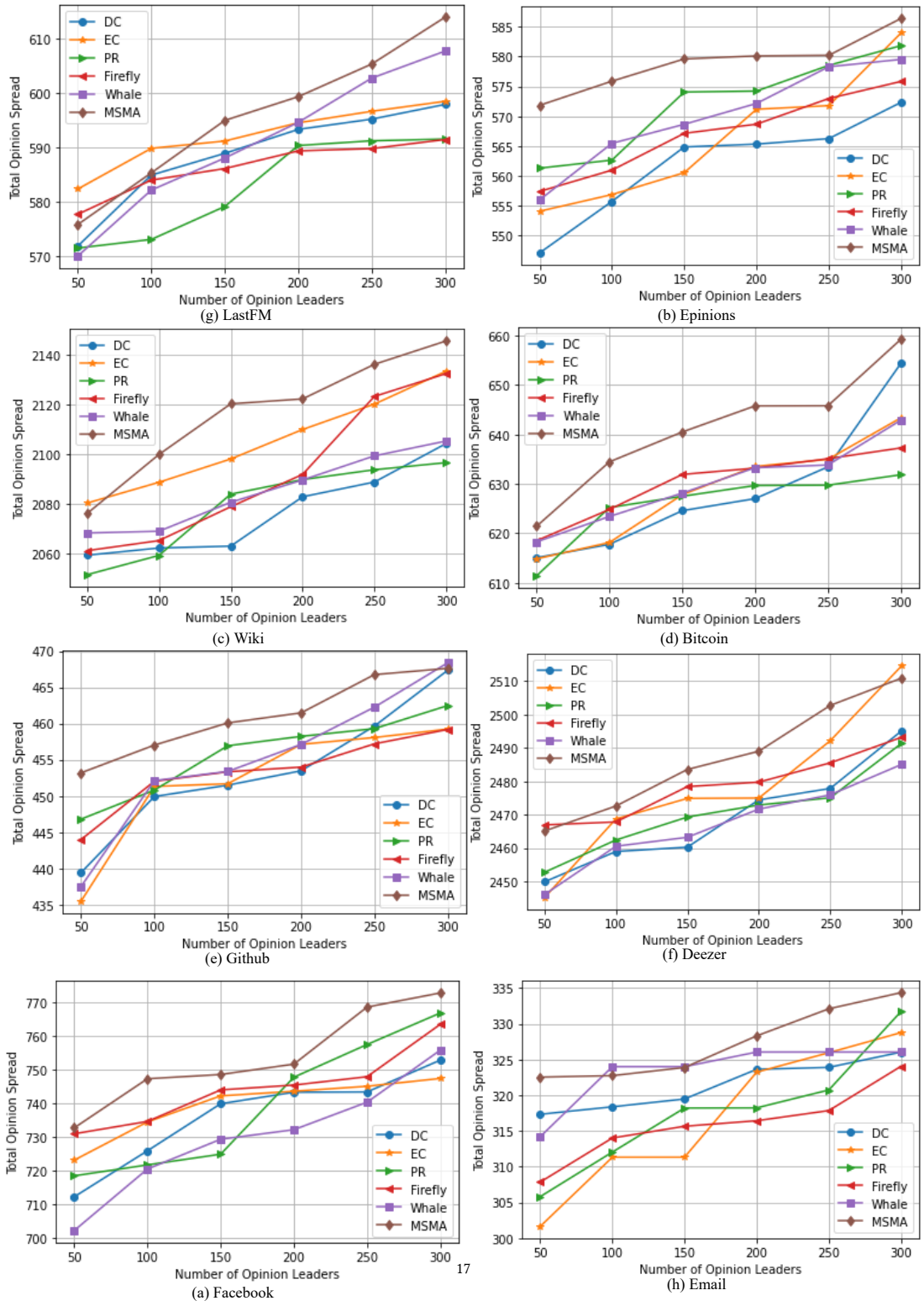


Fig. 4. (a)-(h) Total opinion spread vs. number of opinion leaders value obtained by various methods on different real-life datasets. The simulations were run 100 times under the COD information diffusion model.

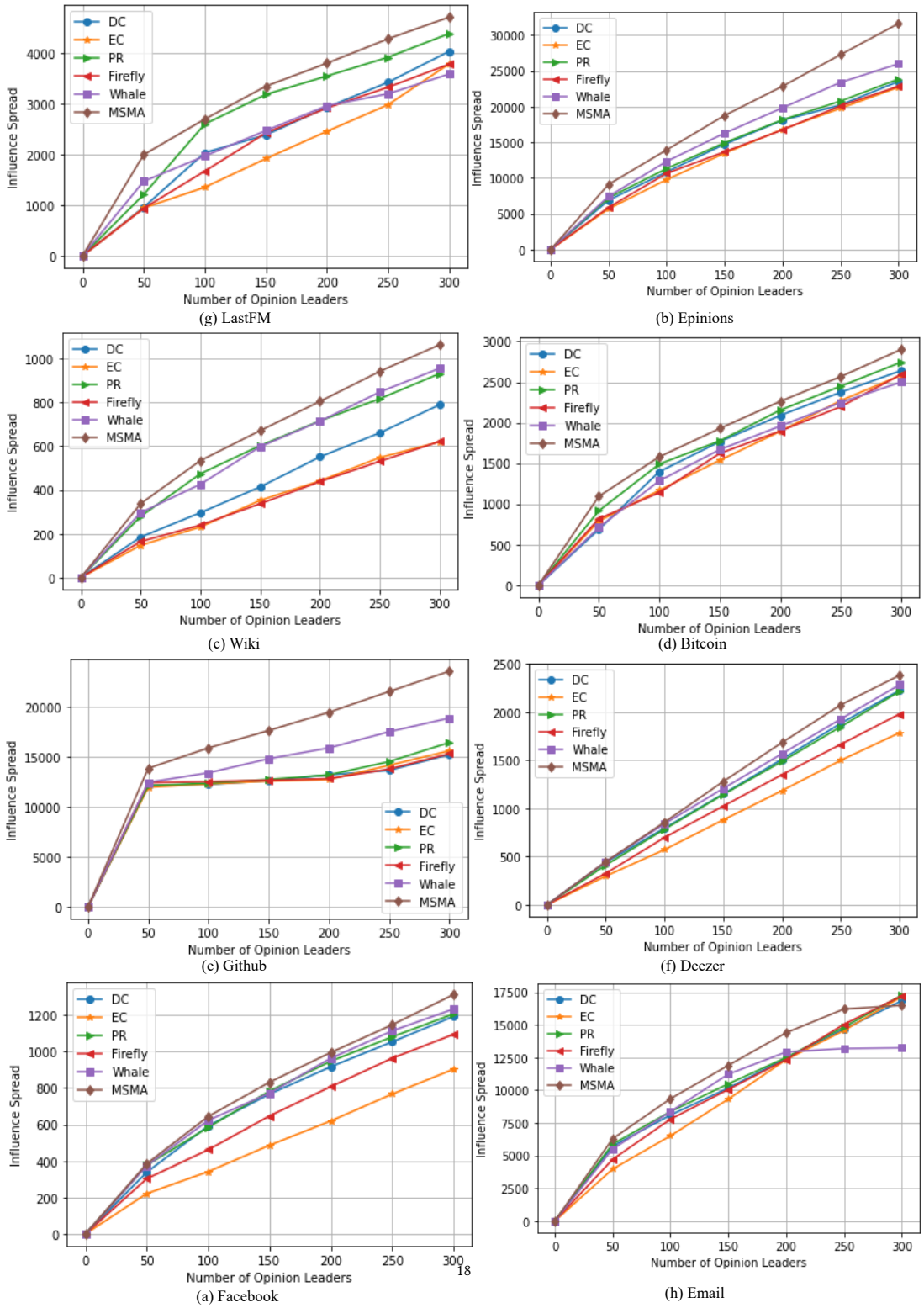


Fig. 5. (a)-(h) Final influence spread vs. number of opinion leaders value obtained by various methods on different real-life datasets. The infection probability, p is set to 0.25 under the IC model. Here, DC, EC, PR, Firefly, Whale, and MSMA refers to Degree Centrality, Eigenvector Centrality (EC), PageRank Centrality (PR), Firefly Optimization, Whale Optimization, and Modified Spider Monkey Algorithm (MSMA), respectively.

spreading process. It can be observed that for Facebook network (fig. 5(a)), Bitcoin network (fig. 5(d)), Deezer network (fig. 5(f)) and the Email network (fig 5(h)) there is a close competition between our proposed Modified Spider Monkey Algorithm (MSMA), the whale optimization based approach and the PageRank (PR) algorithm. But still our proposed approach performs better than these algorithms and rest other algorithms as well. As for the Epinions network (fig. 5(b)), the Wiki network (fig. 5(c)), the Github network (fig. 5(e)) and the LastFM network (fig. 5(g)) our proposed algorithm clearly outperforms rest of the algorithms by a huge margin. For these networks also the Whale optimization based approach and the PageRank approach are close performers to our proposed MSMA approach. Rest of the algorithms follow thus. The above discussion shows that our proposed approach MSMA generates the best quality of opinion leaders for varying number of opinion leaders chosen.

(iii) Susceptible-Infected-Recovered (SIR) model:

Fig. 6 shows the influence spread achieved by various algorithms as the initial number of chosen opinion leaders is varied under the information diffusion model. The infection probability β is set as 0.25 for the SIR model. It can be clearly seen from fig. 6 that the proposed MSMA approach is the best performer across varying number of chosen opinion leaders for all the datasets. PageRank is the second best performer followed by whale optimization approach. The eigenvector centrality (EC) is the worst performer amongst all the compared algorithms. It can also be seen that as the number of initial opinion leaders increases the performance of our proposed algorithm becomes more and more exemplified as the difference between the achieved influence spread by our algorithm and the other algorithms increases greatly for all the datasets. This discussion corroborates the performance standards of our proposed approach.

6.3 Influence Spread vs. Infection Probability

Now we go on to consider the effect of varying the infection probability on the influence spread achieved by various algorithms. The infection probability refers to the probability with which an activated node can infect its neighbors. The infection probability is very critical in IC and SIR information diffusion model. For the IC and the SIR model, the higher the infection probability, the higher will be the achieved influence spread. The analysis of the effect of varying this infection probability for the IC and the SIR model is described below:

(i) Independent Cascade (IC) model:

The effect of varying the infection probability on the influence spread achieved by various algorithms under the IC model can be seen in fig. 7. The number of opinion leaders is chosen to be 100 for all the networks. It can be interpreted from the fig. 7 that the influence spread increases with an increase in infection probability for the IC model. This can be attributed to the fact that with an increase in infection probability, the ability of active nodes to activate their neighboring nodes also increases. It can be seen that for all the networks, the proposed Modified Spider Monkey Algorithm (MSMA) is the best performer. For the Epinions network (Fig. 7(b)), the Wiki network (fig. 7(c)) and the Github network (fig. 7(e)) our proposed approach gives exemplary performance and the influence spread achieved is much better than any algorithm. For the LastFM network (fig. 7(g)), our algorithms performs best until the infection probability is increased to 0.2 but as soon as the infection probability reaches 0.5 then the PageRank algorithm is the best performer. Throughout all the networks,

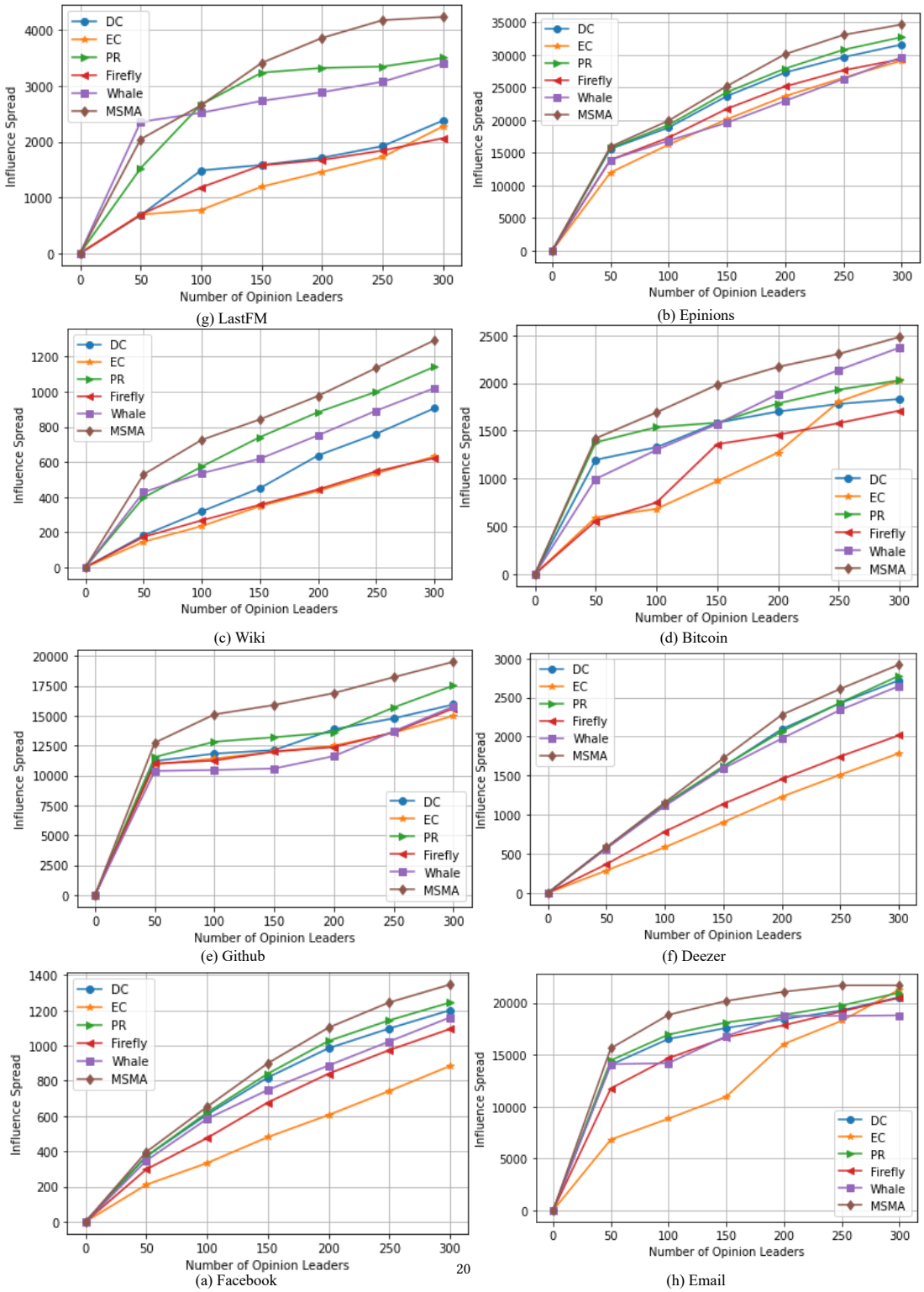


Fig. 6. (a)-(h) Final influence spread vs. number of opinion leaders value obtained by various methods on different real-life datasets. The infection probability, β is set to 0.25 under the SIR model. Here, DC, EC, PR, Firefly, Whale, and MSMA refers to Degree Centrality, Eigenvector Centrality (EC), PageRank Centrality (PR), Firefly Optimization, Whale Optimization, and Modified Spider Monkey Algorithm (MSMA), respectively.

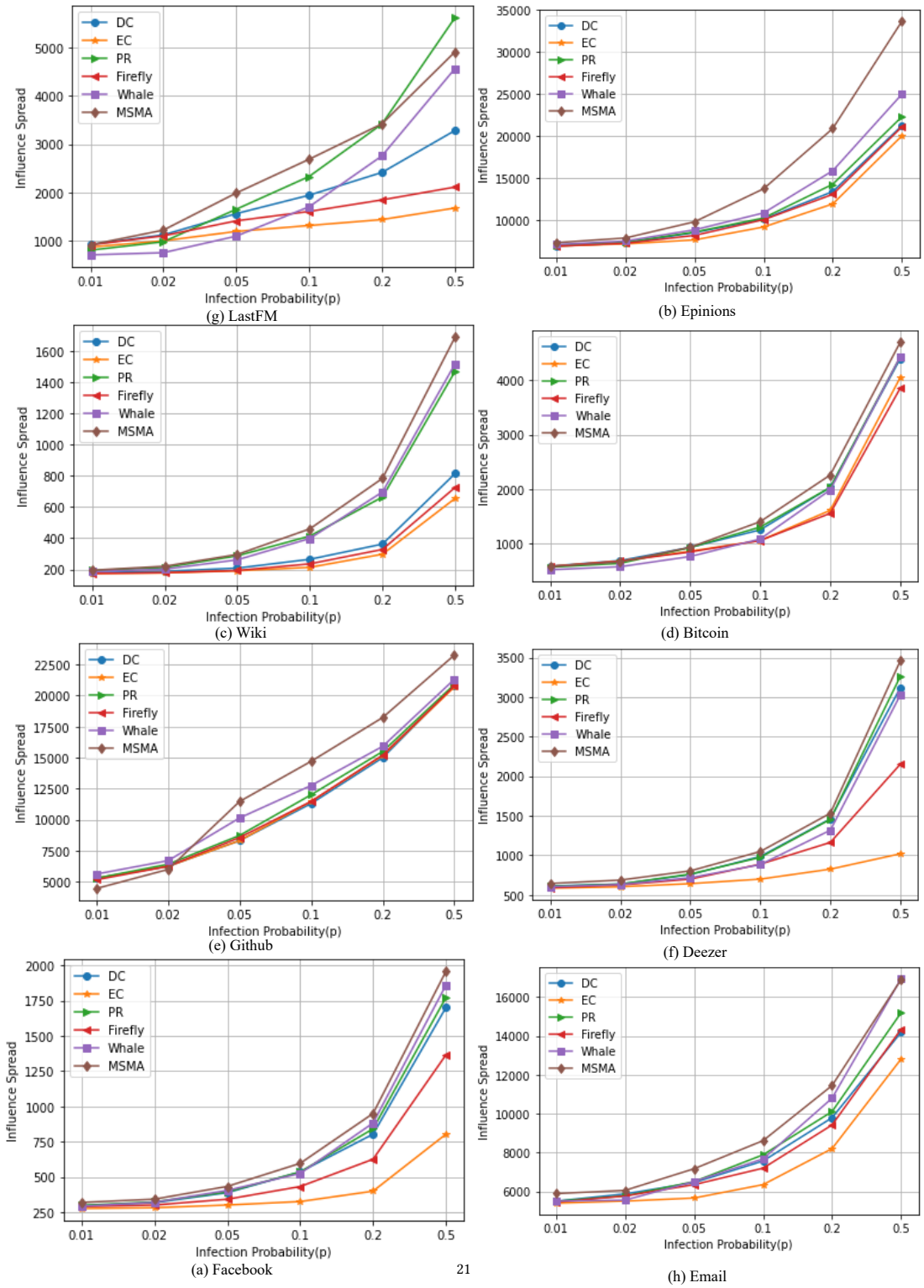


Fig. 7. (a)-(h) Final influence spread vs. infection probabilty, p value obtained by various methods on different real-life datasets. The number of opinion leaders are chosen to be 100 under the IC model. Here, DC, EC, PR, Firefly, Whale, and MSMA refers to Degree Centrality, Eigenvector Centrality (EC), PageRank Centrality (PR), Firefly Optimization, Whale Optimization, and Modified Spider Monkey Algorithm (MSMA), respectively.

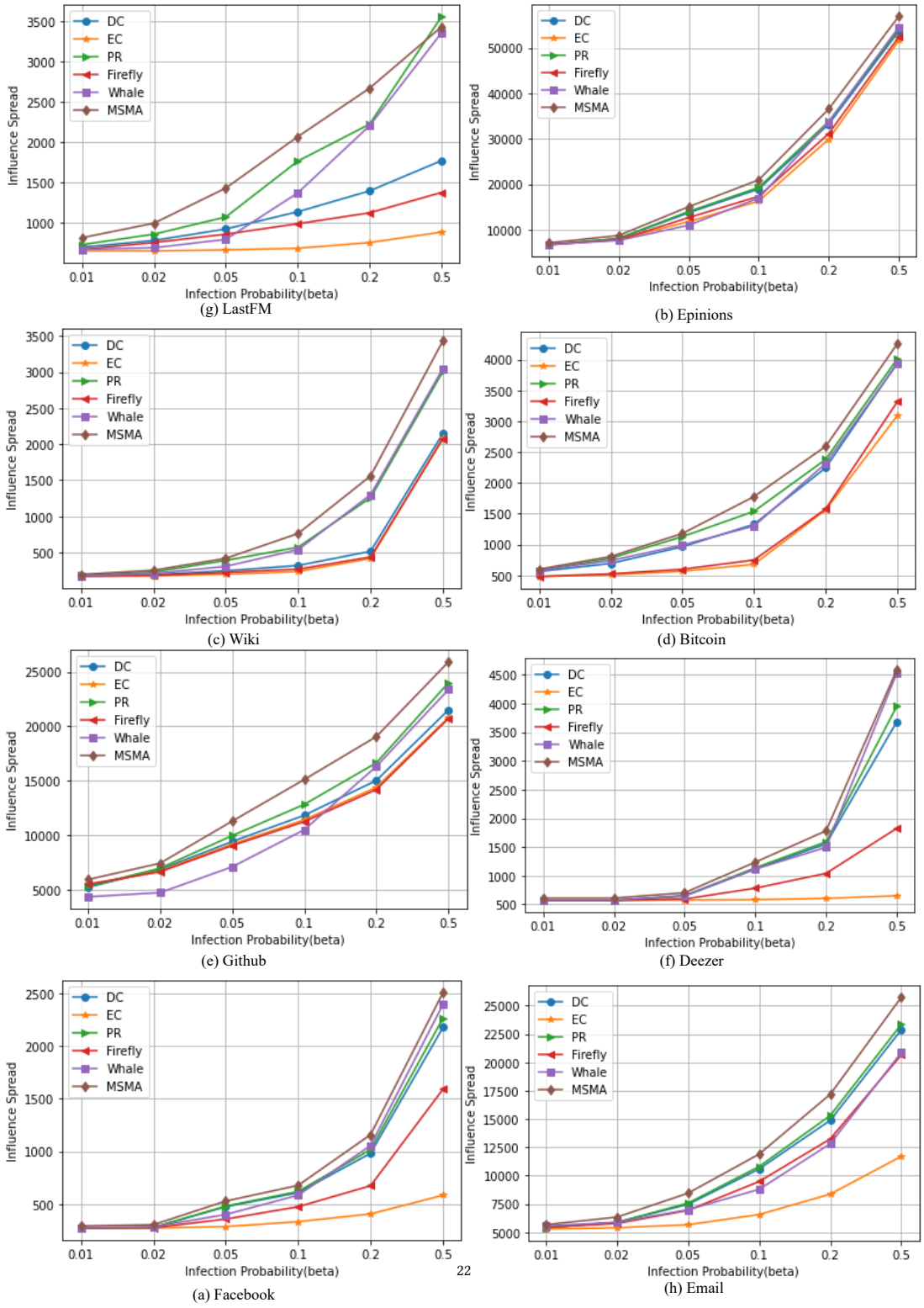


Fig. 8. (a)-(h) Final influence spread vs. infection probabilty, β value obtained by various methods on different real-life datasets. The number of opinion leaders are chosen to be 100 under the SIR model. Here, DC, EC, PR, Firefly, Whale, and MSMA refers to Degree Centrality, Eigenvector Centrality (EC), PageRank Centrality (PR), Firefly Optimization, Whale Optimization, and Modified Spider Monkey Algorithm (MSMA), respectively.

eigenvector centrality (EC) is the worst performer. In almost all of the networks our algorithm performs best followed by whale optimisation based approach and the PageRank approach.

(iii) Susceptible-Infected-Recovered (SIR) model:

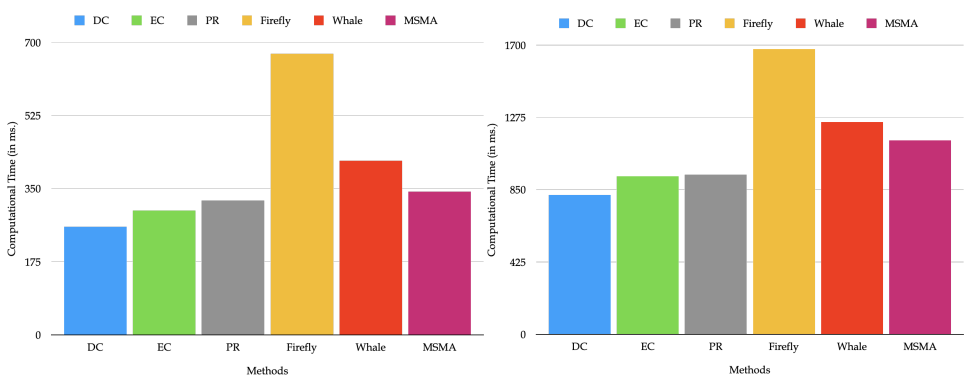
The effect of varying the infection probabolity (β) on the influence spread achieved by various algorithms over all the datasets under the SIR information diffusion model can be visualised in fig. 8. The number of chosen opinion leaders for the experiments were considered to be 100. It can be clearly seen from fig. 8 that as the infection probability (β) increases, the amount of influence spread achieved by various algorithms also increases. It can be observed that for the Facebook network (fig. 8(a)), Epinions network (fig. 8(b)) and the Deezer network (fig. 8(f)) the influence spread achieved by all the algorithms is very close to each other. But even in such close competition, our proposed algorithm MSMA, is the best performer. For the Wiki network (fig. 8(c)), the Bitcoin network (fig. 8(d)), the Github network (fig. 8(e)), the LastFM network (fig, 8(g)) and the Email network (fig. 8(h)), our proposed algorithm clearly outperforms other algorithms by a huge margin. Our algorithm is followed by whale optimised approch and the PageRank (PR) approach in almost all the networks. The Eigenvector Centrality (EC) based approach is the worst performer throughout the networks. The above discussion highlights the exemplary performance of our proposed Modified Spider Monkey Algorithm (MSMA) in a more reliable sense.

6.4 Absolute Computational Time

This section presents a comparison of the computational time required by various algorithms to execute. Fig. 9 represents the time taken by various algorithms to detect 100 opinion leaders for different networks. Since the metaheuristic algorithms are stochastic in nature, so we run 100 trials for every algorithm on every dataset and then average out the results. It can be clearly seen that the degree cetrality based approach for opinion leader detection takes the least amount of time. This can be explained by the fact that degree centrality based approach is a simple heuristic that considers only the degree of a node. While it can also be interpreted from the fig. 9 that the metaheuristic based approaches are the most time consuming. This can be understood by notion that the metaheuristic approaches consider a lot of parameters which are more specific to the problem under consideration and hence they take longer to execute. But even for the various metaheuristic algorithms compared by us for this study, our proposed Modified Spider Monkey Algorithm (MSMA) based approach is the best performer in terms of computational time. Moreover, the performance edge obtained by our MSMA approach in terms of the influence spread achieved, more than compensates for the slightly longer run-time. The above analysis shows the plausibility of our proposed Modified Spider key Algorithm for the purpose of opinion leader detection in terms of computational time requirements as well as the influence spread targeted.

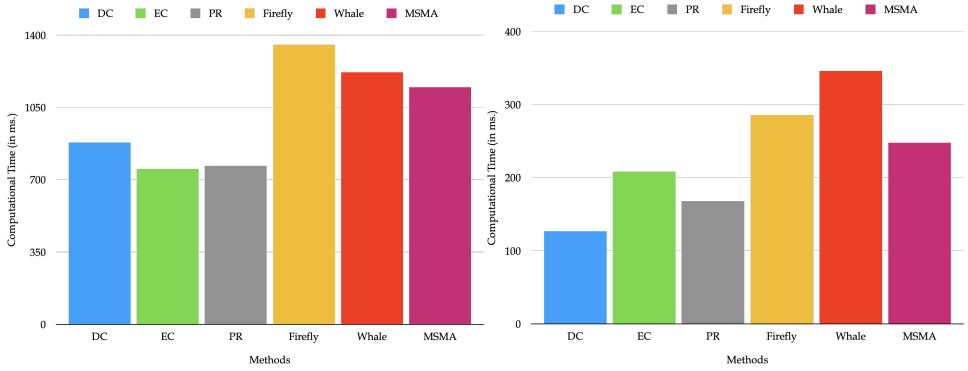
7 CONCLUSION

Social networking has gained immense popularity in the Asia-Pacific region, where billions of active users share their information and opinions. Opinion leaders tend to be the nodes in such networks which tend to have strong impact on other nodes and can dictate the opinion of a particular group in a specific direction. Through this work, we proposed an opinion detection technique in Asian social networks by leveraging the modified node2vec embedding and the spider monkey algorithm based metaheuristic. First of all, we generated the modified node2vec embedding for every node in network, this helps in generating the feature vectors for every node in the network. Then we passed the nodes along with their generated feature vectors to modified spider monkey algorithm. The classical spider monkey



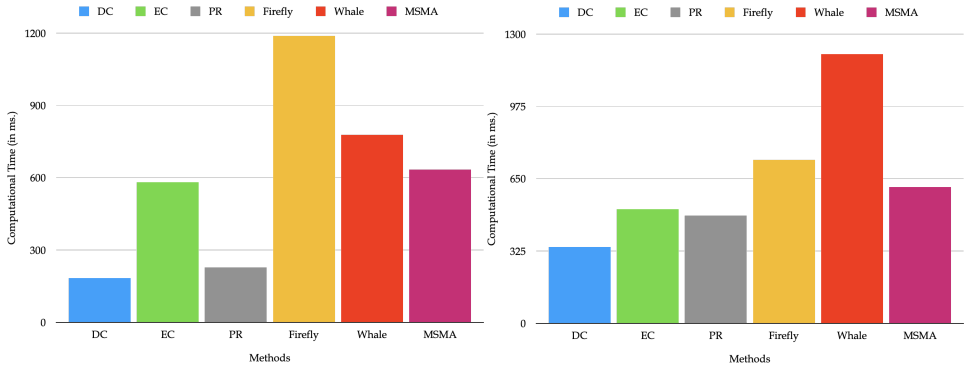
(a) Facebook

(b) Epinions



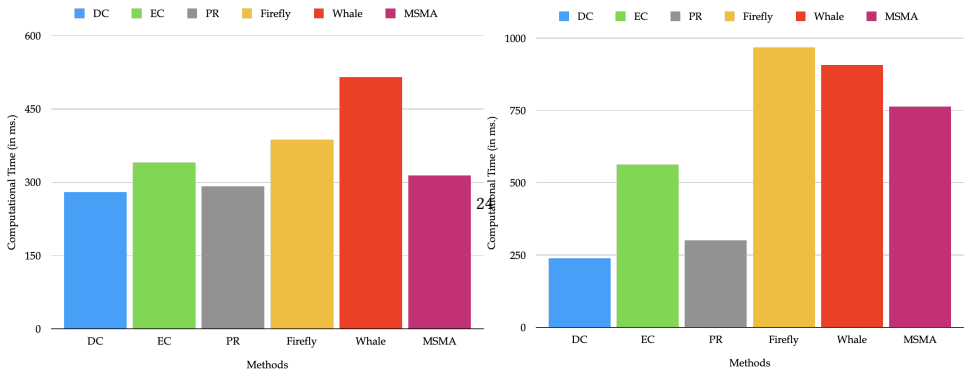
(c) Wiki

(d) Bitcoin



(e) Github

(f) Deezer



(g) LastFM

(h) Email

Fig. 9. (a)-(h) Comparison of the computational time required for various algorithms to select 100 opinion leaders for different datasets. The results are averaged over 100 runs.

algorithm is modified to better fit the opinion leader detection task. We considered the nodes of the network as the spider monkeys for our study. The global and local leaders are selected by using the k-means++ algorithm instead of using the greedy approach. The chosen set of local leaders form the opinion leaders for our task. We performed intensive experimentations on several real-life social networks and compared the performance of our proposed approach by some of the classical algorithms and some of the contemporary metaheuristic algorithms for opinion leader detection. The experiments were run on Independent Cascade (IC), Susceptible-Infected-Recovered (SIR) information diffusion model. The experiments performed revealed the credibility of our proposed approach for the task of opinion leader detection as it generates a great quality of the opinion leaders. As part of the future work, we can try several different metaheuristic based approaches. We can also a hybrid mix of different metaheuristic approaches stacked together.

REFERENCES

[1] Eric Abrahamson and Lori Rosenkopf. 1997. Social network effects on the extent of innovation diffusion: A computer simulation. *Organization science* 8, 3 (1997), 289–309.

[2] Samad Mohammad Aghdam and Nima Jafari Navimipour. 2016. Opinion leaders selection in the social networks based on trust relationships propagation. *Karbala International Journal of Modern Science* 2, 2 (2016), 88–97.

[3] Sameer Anand, Abhishek Mallik, Sanjay Kumar, et al. 2022. Integrating node centralities, similarity measures, and machine learning classifiers for link prediction. *Multimedia Tools and Applications* (2022), 1–29.

[4] Olivier Bachem, Mario Lucic, S Hamed Hassani, and Andreas Krause. 2016. Approximate k-means++ in sublinear time. In *Thirtieth AAAI conference on artificial intelligence*.

[5] E Bakshy, I Rosenn, C Marlow, and L Adamic. 2012. April 16-20. The role of social networks in information diffusion. *Proceedings of ACM WWW* (2012).

[6] Jagdish Chand Bansal, Harish Sharma, Shimpi Singh Jadon, and Maurice Clerc. 2014. Spider monkey optimization algorithm for numerical optimization. *Memetic computing* 6, 1 (2014), 31–47.

[7] Mikhail Belkin and Partha Niyogi. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering.. In *Nips*, Vol. 14. 585–591.

[8] Freimut Bodendorf and Carolin Kaiser. 2010. Detecting opinion leaders and trends in online communities. In *2010 Fourth International Conference on Digital Society*. IEEE, 124–129.

[9] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems* 30, 1-7 (1998), 107–117.

[10] Kenny K Chan and Shekhar Misra. 1990. Characteristics of the opinion leader: A new dimension. *Journal of advertising* 19, 3 (1990), 53–60.

[11] Yi-Cheng Chen, Lin Hui, Chun I Wu, Hsin-Yu Liu, and Sheng-Chih Chen. 2017. Opinion leaders discovery in dynamic social network. In *2017 10th International Conference on Ubi-media Computing and Workshops (Ubi-Media)*. IEEE, 1–6.

[12] Youngsang Cho, Junseok Hwang, and Daeho Lee. 2012. Identification of effective opinion leaders in the diffusion of technological innovation: A social network approach. *Technological Forecasting and Social Change* 79, 1 (2012), 97–106.

[13] Renee Dye. 2000. The buzz on buzz. *Harvard business review* 78, 6 (2000), 139–146.

[14] David Gold. 1956. Personal Influence: The Part Played by People in the Flow of Mass Communications.

[15] Amit Goyal, Francesco Bonchi, and Laks VS Lakshmanan. 2008. Discovering leaders from community actions. In *Proceedings of the 17th ACM conference on Information and knowledge management*. 499–508.

[16] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[17] Derek L Hansen, B Shneiderman, MA Smith, and I Himelboim. 2011. Social network analysis: measuring, mapping, and modeling collections of connections In:"Analyzing Social Media Networks with NodeXL: Insights from a Connected World".".

[18] Julia Heidemann, Mathias Klier, and Florian Probst. 2010. Identifying key users in online social networks: A pagerank based approach. (2010).

[19] Lokesh Jain and Rahul Katarya. 2018. Identification of opinion leader in online social network using fuzzy trust system. 233–239. https://doi.org/10.1109/IADCC.2018.8692095

[20] Lokesh Jain and Rahul Katarya. 2019. Discover opinion leader in online social network using firefly algorithm. *Expert systems with applications* 122 (2019), 1–15.

[21] Lokesh Jain, Rahul Katarya, and Shelly Sachdeva. 2020. Opinion leader detection using whale optimization algorithm in online social network. *Expert Systems with Applications* 142 (2020), 113016.

[22] Bryan Klimt and Yiming Yang. 2004. Introducing the Enron corpus.. In *CEAS*.

[23] Sanjay Kumar and Rahul Hanot. 2020. Community detection algorithms in complex networks: A survey. In *International Symposium on Signal Processing and Intelligent Recognition Systems*. Springer, 202–215.

[24] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and VS Subrahmanian. 2018. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 333–341.

[25] Sanjay Kumar, Abhishek Mallik, Anavi Khetarpal, and BS Panda. 2022. Influence maximization in social networks using graph embedding and graph neural network. *Information Sciences* (2022).

[26] Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. 2016. Edge weight prediction in weighted signed networks. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 221–230.

[27] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. 2010. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*. 641–650.

[28] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. 2010. Signed networks in social media. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 1361–1370.

[29] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. 2009. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics* 6, 1 (2009), 29–123.

[30] Feng Li and Timon C Du. 2011. Who is talking? An ontology-based opinion leader identification framework for word-of-mouth marketing in online social blogs. *Decision support systems* 51, 1 (2011), 190–197.

[31] Yanyan Li, Shaoqian Ma, Yonghe Zhang, Ronghuai Huang, et al. 2013. An improved mix framework for opinion leader identification in online learning communities. *Knowledge-Based Systems* 43 (2013), 43–51.

[32] Ning Ma and Yijun Liu. 2014. SuperedgeRank algorithm and its application in identifying opinion leader of online public opinion supernetwork. *Expert Systems with Applications* 41, 4 (2014), 1357–1368.

[33] Vincent Mak. 2008. The emergence of opinion leaders in social networks. *Available at SSRN 1157285* (2008).

[34] Julian J McAuley and Jure Leskovec. 2012. Learning to discover social circles in ego networks.. In *NIPS*, Vol. 2012. Citeseer, 548–56.

[35] James H Myers and Thomas S Robertson. 1972. Dimensions of opinion leadership. *Journal of marketing research* 9, 1 (1972), 41–46.

[36] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 1105–1114.

[37] Tiago Pimentel, Adriano Veloso, and Nivio Ziviani. 2018. Fast node embeddings: Learning ego-centric representations. (2018).

[38] Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. 2003. Trust management for the semantic web. In *International semantic Web conference*. Springer, 351–368.

[39] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-scale attributed node embedding. *Journal of Complex Networks* 9, 2 (2021), cnab014.

[40] Benedek Rozemberczki and Rik Sarkar. 2020. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1325–1334.

[41] Michael Trusov, Anand V Bodapati, and Randolph E Bucklin. 2010. Determining influential users in internet social networks. *Journal of marketing research* 47, 4 (2010), 643–658.

[42] Ralf Van der Merwe and Gene Van Heerden. 2009. Finding and utilizing opinion leaders: Social networks and the power of relationships. *South African Journal of Business Management* 40, 3 (2009), 65–76.

[43] Daniele Vilone, Francesca Giardini, Mario Paolucci, and Rosaria Conte. 2016. Reducing individuals' risk sensitiveness can promote positive and non-alarmist views about catastrophic events in an agent-based simulation. *arXiv preprint arXiv:1609.04566* (2016).

[44] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 1225–1234.

[45] Li Yang, Yafeng Qiao, Zhihong Liu, Jianfeng Ma, and Xinghua Li. 2018. Identifying opinion leader nodes in online social networks with a new closeness evaluation algorithm. *Soft Computing* 22, 2 (2018), 453–464.

[46] Xin-She Yang. 2020. *Nature-inspired optimization algorithms*. Academic Press.

[47] Maoran Zhu, Xingkai Lin, Ting Lu, and Hongwei Wang. 2016. Identification of opinion leaders in social networks based on sentiment analysis: Evidence from an automotive forum. *Advances in Computer Science Research* 58 (2016), 412–416.